Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Rozpoznávání a detekce pohybů pomocí senzoru Microsoft Kinect

Object Recognition Using Microsoft Kinect

2014 **Bc. Martin Brzoza**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Brzoza**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Rozpoznávání a detekce pohybů pomocí senzoru Microsoft Kinect
Object Recognition Using Microsoft Kinect

Zásady pro vypracování:

Detekce pohybů a rozpoznávání patří dnes k velmi rozšířeným algoritmům využívajících se např. v oblastech bezpečnostních aplikací.
Cílem této práce je zaměřit se na možností rozpoznávání a detekci pohybů řidiče v autě pomocí senzoru Microsoft Kinect.

1. Seznamte se s algoritmy a postupy pro problematikou rozpoznávání a detekci pohybů. Zaměřte se na rozpoznávání pomocí senzoru Microsoft Kinect (využití hloubkové mapy).
2. Naimplementujte a otestujte aplikaci umožňující detekovat a rozpoznávat pohyby lidského těla (hlava, trup, ramena lokty, ruce) řidiče jedoucího v autě.
3. Výsledky své práce prakticky otestujte, srovnejte a vyhodnoťte.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Martin Němec, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *5. května 2014*

.....
podpis studenta

Poděkování

Rád bych poděkoval Ing. Martinu Němcovi, Ph.D za odbornou pomoc a konzultaci při vytváření této diplomové práce a také všem, kteří mi byli podporou při práci a mají na výsledku nemalý podíl.

Abstrakt

V textu je rozebrána současná situace na poli detekce pohybu postavy člověka pomocí zařízení Kinect a generování její kostry. Situace je konkretizována na postavu řidiče v automobilu. V současné době není tato oblast, věnující se dané problematice, veřejně příliš probádaná. Většina aplikací věnující se generování kostry je optimalizována pro stojící postavy a prostředí, kde se v oblasti postavy nenalézají žádné nadbytečné předměty. Proto při zasazení do našeho případu dochází ke značným deformacím výsledků. V textu je proto navržena korektura současných metod pomocí odstranění statických předmětů modelem pozadí a vylepšení detekce úpravou algoritmů pro detekci dlaní a loktů. Jako výchozí knihovna pro detekci kostry je použita otevřená knihovna Skeltrack pracující pod ovladači OpenKinect.

Klíčová slova

Kinect, detekce kostry, OpenKinect, skeltrack

Abstract

In this thesis is discussed current situation at field of person movement recognition using Microsof Kinect device and skeleton tracking. Situation is concretized on driver in car. Currently this area is not publicly known well. Most of applications for skeleton tracking using Kinect are optimized for standing person and environment where are not so many objects surrounding tracked person like steering wheel. This is the reason why current skeleton tracking applications fails. In this text correction of current methods is presented with help of static background removing and improved tracking algorithms for elbows and hands. As default library for skeleton tracking “Skeltrack” is used. OpenKinect drivers are used for communication with Kinect device.

Key words

Kinect, skeleton tracking, OpenKinect, skeltrack

Obsah

Úvod

Oblast počítačového vidění od uvedení zařízení Kinect na trh zaznamenala obrovský vývoj kupředu. Od splnění počátečního účelu v podobě vysoce interaktivních her si mnoho lidí uvědomilo mnohem vyšší potenciál v podobě zařízení, které díky nově poskytovanému hloubkovému obrazu ulehčuje některé disciplíny počítačového vidění. Není již třeba se jen spoléhat na barevnou informaci klasických kamer, která je lehce ovlivnitelná změnou osvětlení. Začaly se objevovat pokusné aplikace, které mají za účel snímat lidské pohyby a promítat je na 3D počítačové modely a tím nahrazovat drahé motion capture zařízení. Nemalé uplatnění je možno nalézt v robotice, kde díky snadnější segmentaci objektů ve scéně je možno vytvářet inteligentnější algoritmy pro pohyb kolem překážek. Existují také komerční aplikace, které dokáží ze scén vytvářet 3D modely.

V mé práci budu snažit navázat na aplikace analýzy pohybu postav ve scéně. Tyto aplikace generují na základě pohybující se postavy její kostru a klient má v takovém případě k dispozici pozice různých hlavních lidských kloubů pohybující se postavy. Aplikace bude konkretizována na případ postavy řidiče automobilu. Takovéto řešení pak může mít praktické využití v automobilovém průmyslu, kde je třeba analyzovat postavu řidiče a její pohyby. Výstupy aplikace je možno využít například pro ovládání zařízení v automobilu pomocí gest. Součástí implementace je také návrh sady gest a jejich detekce. Nemalé uplatnění může aplikace nalézt také při zvyšování bezpečnosti v automobilu. Řídící systémy automobilu mohou podle konkrétních pozic částí těla řidiče regulovat výkony airbagu apod.

Struktura textu je členěna do několika kapitol. V prvních kapitolách se čtenář dočte o aktuálním stavu nasazení zařízení do prostředí automobilu. Následuje kapitola pokrývající teoretický základ o zařízení Kinect. Další kapitoly se již věnují členění základních metod pro generování kostry postavy a jejich teoretický základ. Také jsou popsány hlavní aplikace zabývající se detekcí kostry v obraze. Tím je završena část potřebných teoretických základů pro pokračování v dané problematice. Následuje praktická část, kde v její první polovině je nastíněn návrh řešení a případné komplikace následovaný kapitolou věnovanou implementaci. Text je završen informacemi o testování a zhodnocením uvedeném v poslední kapitole.

1 Aktuální stav

Vzhledem k tomu, že se v mé práci budu věnovat možnosti zavedení zařízení Kinect do automobilu, je nutné nastínit aktuální stav v této oblasti. V současné době již existují zavedené pojmy jako AutoNUI, čili možnosti přirozeného uživatelského rozhraní přeneseného do prostředí automobilu. Probíhají stejnojmenné semináře [1], které mají za úkol vymezit základní principy a možnosti této oblasti. Jejími cíli jsou například prozkoumání zavedení ovládání gest do prostředí automobilu, jak pro řidiče, tak pasažéry. Nalezení dopadu na komfort řidiče a určení celkového uživatelského rozhraní. V [1] je také zmíněno, že takový systém by měl být i určitým způsobem zábavný a příjemný, aby byl cílovou skupinou dobře přijat, přeci jen někteří řidiči tráví v automobilu podstatnou část dne a je cílem jim tyto chvíle ulehčit a zpříjemnit.

Jedním z prvních průkopníků zavedení zařízení Kinect do automobilu byla společnost Microsoft v projektu Detroit[2]. Kinect zde byl instalován do zadní a přední části Fordu Mustang a možnosti využití byly spíše pro sledování a analyzování chodců a překážek. V budoucnu má však najít využití i v přístrojové desce, kde by umožňoval ovládání zařízení gesty. Nabídnul tak přepínání informačních panelů, aniž by příliš zvýšil rozptylování řidiče.

Dalším zástupcem je prototyp modelu Smart Insect společnosti Toyota[3]. Zabudovaný senzor umožňuje rozpoznat uživatele na základě tvaru těla a analýze obličeje. Při rozpoznání je zobrazeno přivítání rozpoznaného řidiče. Také umožňuje predikci chování řidiče na základě jeho pohybů a na tomto základě například otevřít dveře. Jsou také využívány možnosti rozpoznávání hlasu a ovládání příkazy.

Využití našel také Gibson Hu, student technické univerzity v Sydney[4]. Využil zabudovaný Kinect v zadním blatníku automobilu, aby umožnil na přístrojové desce přenášet data pro asistované parkování. Jeho systém přenáší barevný, infračervený obraz a také obsahuje prezentaci v určitém režimu ptačího pohledu.

Jak je vidět, tato oblast nasazení je ještě docela čerstvá a veřejně ne příliš probádaná. Proto je určitě velkou výzvou, ale zároveň velkou otázkou. Nikdo neví, jak bude ve výsledku cílovou skupinou přijata a jaký bude mít výsledný dopad na komfort a bezpečnost řízení. Bude tedy ještě nejspíš nějakou dobu trvat, než přejde z fáze prototypu do reálného nasazení.

2 Kinect

Zařízení bylo představeno v listopadu roku 2010[5] jako příslušenství herní konzole Xbox 360. Získaná data kombinovala geometrii a vizuální atributy. Díky tomu se Kinect stal flexibilní nástroj použitelný v mnoha oblastech a aplikacích pro počítačovou grafiku, zpracování obrazu a počítačové vidění.

Kinect obsahuje RGB kameru, infračervený zářič a další kameru. Tím je schopné zařízení snímat barevný obrázek a hloubku každého pixelu ve scéně. Tyto data tak reprezentují barevnou a geometrickou informaci o každém pixelu ve scéně. To nám umožňuje dělat úlohy, které byli dříve zvládnutelné mnohem obtížněji s pouhou barevnou informací. Nyní můžeme využívat hloubku, normálu, luminanci apod. Zavádíme nový pojem RGBD obrázku. Jedná se o barevný obrázek obsahující navíc hloubkovou informaci o každém pixelu.

Hloubkový senzor použitý v zařízení byl vyvinut Zeevem Zalevskym, Alexanderem Shpuntem, Aviadem Maizelsem a Javierem Garciou roku 2005 a vydání zařízení bylo oficiálně oznámeno 1. ledna 2005 pod projektem Natal na výstavě E3. Natal je odvozen od stejnojmenného brazilského města. Původním záměrem bylo vytvořit nástroj umožňující uživateli komunikovat pomocí gest a hlasu s platformou Xbox 360. Toto je důvod proč byl Kinect schopen snímat pouze v rozlišení 640x480 a 30Hz, protože výkonost hrací konzole v té době nedovolovala nasazení vyšších rozlišení. Spolu s hloubkovými daty je možno díky dostupnému software získat kostru osoby pohybující se před senzorem. Spolu s touto kostrou je možno definovat uživatelské gesta. V prvních 60 dnech od uvedení zařízení Kinect bylo prodáno více než 8 miliónů kusů. Díky tomu bylo zařízení zapsáno do Guinessovy knihy, jako nejrychleji se prodávající konzumní zařízení. V červnu roku 2011 společnost Microsoft vydala SDK pro systém Windows 7. V únoru 2012 pak byl spuštěn projekt zařízení Kinect pro systém Windows.

Obrázek 1.1: *Zařízení Kinect [5]*

Chronologie dostupných nástrojů pro vývoj na zařízení Kinect:

- 10. Listopadu 2010 Hector Martin uvolnil jako open source projekt jeho knihovnu OpenKinect/libfreenect

- 9. Října 2010 se společnost PrimeSense rozhodla spustit vlastní open source ovladač a API OpenNI
- 16. Června 2011 byl ve 12 zemích uvolněn společností Microsoft SDK pro operační systém Windows 7, v době tvorby práce již ve verzi 1.8

2.1 Hloubkový snímací systém

Součástí je zářič infračerveného laserového paprsku a infračervená kamera. Laser emituje známý vzor šumu. Infračervená kamera pracuje na frekvenci 30Hz a vytváří obrazy v rozlišení 1200x960 pixelů. Toto rozlišení je pak sníženo na 640x480 s 11 bity, což dává k dispozici 2048 úrovní hloubky. Nicméně USB 2.0 má pro tyto účely relativně malou průchodnost, veškeré Kinect data si vezmou okolo 70% USB hubu pro přenos svých dat. Díky tomu je nemožné pouštět paralelně dvě zařízením Kinect na stejném hubu. Zorné pole je 58 stupňů horizontálně, 45 stupňů vertikálně a 70 stupňů diagonálně. Operační rozsah je mezi 0.8 metry a 3.5 metry. Hloubka je měřena pomocí strukturované světelné metody. Laserem jsou vyzařovány do prostředí tečky o známém vzoru. Tyto tečky jsou pak snímány infračervenou kamerou a porovnávány s tímto známým vzorem. Kvůli tomu, že některé materiály příliš dobře neodrážejí infračervené světlo, je doporučováno snímat objekty okolo 1,8 metrů od senzoru.

2.2 RGB kamera

RGB kamera pracuje také na 30Hz a rozlišení 640x480 a má k dispozici 8 bitů na každý kanál. Kinect má také možnost přepnout tuto kameru do rozlišení 1280x1024 pixelů s rychlostí 10 snímků za sekundu. Kamera má k dispozici několik vlastností, jako automatické vyvážení bílé, referenční černá, omezení blikání, barevné nasycení a opravu defektů.

2.3 Motor, Akcelerometr a série mikrofonů

Zařízení Kinect je vybaveno možností pohybovat svou snímací hlavou nahoru a dolů. Díky akcelerometru je schopno detekovat v jaké pozici se nachází. Pole mikrofonů obsahuje 4 mikrofony operující ve vlastním kanále na 16 bitech a samplovací frekvencí 16kHz.

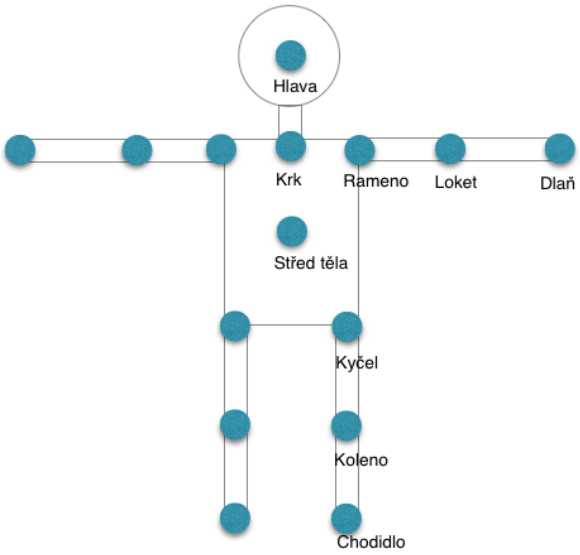
RGBD reprezentace

Jedná se nejpřirozenější data, která můžeme ze zařízení dostat. Jedná se o kombinaci barevných kanálů pro červenou, zelenou, modrou a navíc obdržíme kanál, který obsahuje hloubkovou informaci. Hodnota hloubky je uložena na 11 bitech pro ušetření místa. Hodnota hloubky může také pomocí specifické kalibrace nabývat hodnot reprezentující vzdálenost pixelu od zařízení v milimetrech.

3 Metody detekce kostry

Tato kapitola se věnuje rozboru některých dosud vydaných metod pro detekci kostry ve snímaném obraze. Tyto metody jsou rozděleny do tří částí. Metody založené na hledání schody s modelem bývaly používány většinou v rané fází výzkumů a jsou založené na různých matematických a statistických modelech. Metody založené na strojovém učení používají většinou pro svou detekci množinu anotovaných dat různých osob, které jsou dále použity pro natrénování nějakou z učících se metod. Hybridní metody kombinují prvky obou kategorií.

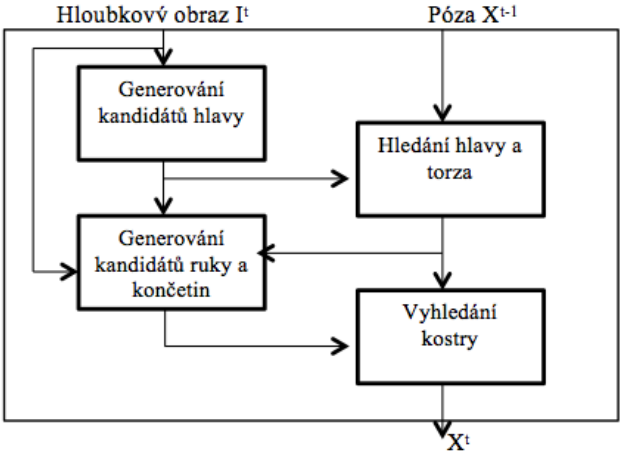
Detekování póz a analyzování postavy, ať již ve statickém či sekvenčním obraze bylo vždy velkou výzvou v oblasti počítačového vidění. Do doby nástupu senzoru Kinect byla tato úloha velmi obtížně realizovatelná pouze s barevnou informací. Současné aplikace využívající Kinect však již dokáží podávat velice dobré výsledky. Výstupem takovýchto aplikací bývá aproximovaná a zjednodušená kostra sledované postavy. Tato výsledná kostra podle konkrétní aplikace může obsahovat pozice jednotlivých hlavních kloubů a částí postavy. Většinou se jedná o pozice hlavy, levého a pravého ramene, loktu a dlaně v případě detekce horní části těla. V případě dolní můžeme detekovat levé a pravé koleno a chodidlo. Příklad takovéto kostry můžeme vidět na obrázku 1.2, tak jak jej reprezentuje například knihovna OpenNI. Pokud se tedy budeme dále odkazovat na název některého z kloubu, budeme vycházet z tohoto obrázku.



Obrázek 1.2: *Reprezentace kostry*

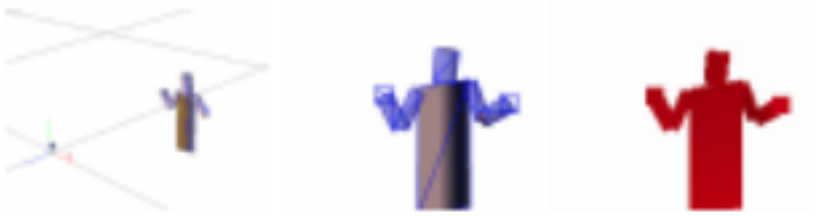
3.1 Metody založené na hledání schody s modelem

Autoři prací využívají většinou nějaké ze statistických metod pro předpověď póz na základě některé z distribucí pravděpodobnosti. Jednou z takových prací je i [6]. Autor zde jako jádro zpracování uvádí datově řízený MCMC Framework. Jedná se o sadu algoritmů budujících pravděpodobnostní distribuci na základě sestaveného Markovova řetězce.



Obrázek 1.3: *MCMC detekce kostry*

Jak je vidět z obrázku výše, do systému vstupuje aktuální hloubkový obraz It a předešlá póza Xt-1. Výstupem je pravděpodobná póza Xt. Vstupem do MCMC frameworku jsou kandidáti hlavy, aby se omezilo prohledávání stavového prostoru. Celá póza je reprezentovaná množinou prostorových cylindrů fixní šířky, které mají zakončení v konkrétním kloubu kostry. Tuto reprezentaci můžeme spatřit na obrázku níže.



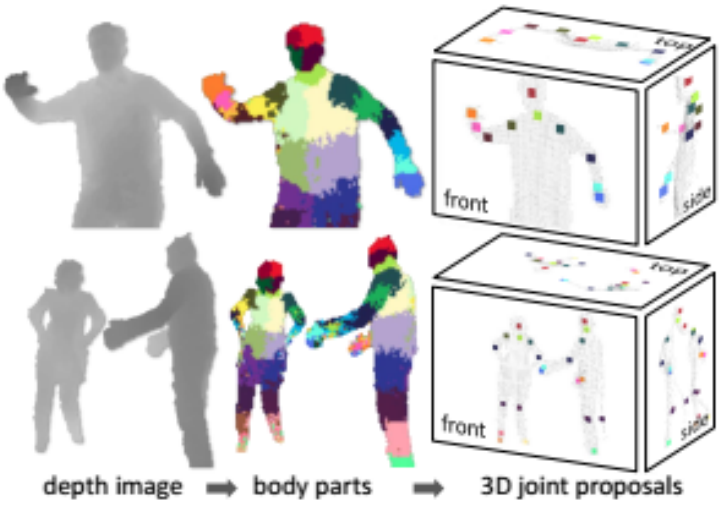
Obrázek 1.4: *Reprezentace kostry pomocí cylindrů[6]*

Pro vyhledání kandidátů hlavy je prohledáván obrys v hloubkovém obrazu pomocí Cannyho hran. Jako nejlepší kandidát je ten, jehož hodnoty nejlépe pokrývají obrys hlavy. Pro vyhledávání kandidátů na ruce jsou nalezeny koncové body ztenčeného profilového obrázku osoby a jsou také vybrány hloubkové body které jsou relativně blízko kamery. Dále je také navrženo několik postupů na segmentování předloktí, kde se vychází z kandidátů na dlaň a hledá se směr natočení předloktí o dané délce.

Práce podává relativně dobré výsledky v případech že jsou obě paže dobře viditelné. Detekce se také těžce vyrovnává s případy, kdy selže některý s detektorů. Také pokud jsou pohyby rukou příliš rychlé, dochází k výpadkům.

3.2 Metody založené na strojovém učení

Jak již bylo řečeno v úvodu, tyto metody ke své detekci často využívají nějakou s učících se metod, pro jejichž trénovací data jsou použity sady anotovaných dat lidských postav. Návrh takovéto metody můžeme nalézt i v práci [7]. Je zde navržena metoda pro rychlou a přesnou predikci kloubů v lidské postavě v jednom hloubkovém obrazu bez použití dočasných informací. Tato metoda stojí určitě za povšimnutí neboť se jedná o stejnou metodu, jež je použitá v systému detekce kostry společnosti Microsoft. Rozpoznávání je rozčleněno do částí, kdy jsou určeny pozice kandidátů na pozice všech kloubů. Hloubkový obraz je segmentován do částí těla s hustotou pravděpodobnosti, které jsou prostorově lokalizované blízko zájmovým kloubům. Re-projekcí odvozených částí do světových souřadnic jsou lokalizovány prostorové módy distribuce každé části a je generováno (někdy i několik) návrhů na 3D pozice každého kloubu, vážené mírou věrohodnosti. Segmentace do částí je vykonávána jako per-pixel klasifikace. Vyhodnocování každého pixelu separátně zamezuje kombinatorickému prohledávání přes rozdílné klouby, ačkoli uvnitř jedné části jsou zde samozřejmě stále dramatické rozdíly v kontextovém vzhledu.



Obrázek 1.5: *Princip fungování systému na detekci kostry[7]*

Částečný princip lze spatřit na obrázku výše.

Pro tréninkové data jsou generovány realistické syntetické hloubkové obrazy lidí různých tvarů a velikostí v široké skále póz získaných databáze „motion capture“. Je natrénován vygenerovaný rozhodovací stromový klasifikátor, který zamezuje přeučení použitím stovek tisíců trénovacích obrazů. Jednoduchá diskriminační hloubková porovnání rysů obrazu produkují nezávislost na 3D posunutí zatímco umožňují vysokou výpočetní výkonnost. Pro další vylepšení výkonnosti může být klasifikátor spouštěn paralelně na GPU pro každý pixel. Prostorové módy odvozených per-pixel distribucí jsou spočítány použitím „mean shift“ jejímž výsledkem jsou návrhy 3D pozic kloubů. Jedním z hlavních přínosů je přístup k detekci pózy, jako rozpoznávání objektu použitím reprezentace předběžných částí těla navržených pro prostorové lokalizovaní kloubů zájmu z nízkými výpočetními nároky a vysokou přesností.

Mnoho technik se snaží překonat nedostatek kvalitních dat. Data založená na intenzitě barev jsou jen velice těžce použitelná kvůli variaci textur a barev oblečení, vlasů apod. Hloubkový obraz tento problém výrazně omezil, ale ještě stále jsme omezeni velkou variací typů a tvarů postav. Jako postavy jsou používány většinou jen siluety ze syntetických modelů, kterým pohyb dodávají „motion capture“ data. Tyto údaje nám umožní vygenerovat velkou sadu trénovacích dat pro různé pózy postavy.

Různé rozsahy pohybů pro „motion capture“ data jsou generovány s ohledem na použití v zábavném průmyslu a jsou vybírány pohyby pro tuto oblast typické. Je také očekáváno, že klasifikátor umí generalizovat neviděné části těla. Do trénovací množiny také nejsou zahrnuty data s variací rotací kolem vertikální osy. Tyto variace lze totiž jednoduše dopočítat. Nejsou také zahrnuty data, kde došlo k nepatrným změnám v pohybu v kloubech. Ze všech těchto požadavků byl vytvořen generátor náhodných dat produkující označené trénovací obrázky. Tento generátor náhodně vygeneruje sadu parametrů a podle nich pak vytváří hloubkové obrazy. Náhodně jsou také používány nastavení jemných změn ve výšce a šířce postavy. Různé natočení kamery apod. Různé části těla jsou pak v těchto datech označeny pomocí textury.

Označení části těla v klasifikaci je dále používáno pro lokalizaci konkrétního kloubu nebo jsou používány pro lokalizaci v kombinaci s jinou částí těla nebo některé jen vyplňují mezeru mezi dalšími částmi těla.

Dále byly navrženy jednoduché příznaky počítané na hloubkovém obraze. Tyto příznaky jsou invariantní vůči 3D posunutí a počítají pouze s jednoduchými aritmetickými operacemi nad hodnotami hloubky. Více se lze dočíst v konkrétním dokumentu. Autoři uvádějí, že tyto příznaky lze jednoduše urychlit přepočítáváním na grafickém procesoru.

Na základě těchto příznaků jsou následně budovány množiny rozhodovacích stromů. Ty se ukázaly jako rychlé a efektivní více třídní klasifikátory pro mnoho úkolů a můžou být efektivně implementovány na GPU. Každá množina je soubor T rozhodovacích stromů, každý skládající se z rozdělujících a listových uzlů. Každý rozdělující uzel se skládá ze zveřejněného příznaku a prahu. Klasifikování pixelu v obrazu je započato v kořenu a ve stromu se postupuje

podle vypočítaných hodnot, kde směr pokračování určuje práh. V listovém uzlu je uložena finální distribuce částí těla.

Každý strom je natrénován na odlišné množině vytvořených obrazů pomocí algoritmu zveřejněného v [7].

Detekce částí těla produkuje per-pixel informaci. Tuto informaci je třeba produkovat napříč všemi pixely k vytvoření informace o konečné 3D pozici kloubu. Tyto pozice jsou finální výsledek algoritmu. Jednoduchým řešením by mohl být výpočet středu objemu jednotlivých částí těla. Namísto toho je použit algoritmus mean shift s váženým jádrem. To umožňuje, aby odhady byly hloubkově invariantní a také zlepšuje přesnost.

3.3 Hybridní metody

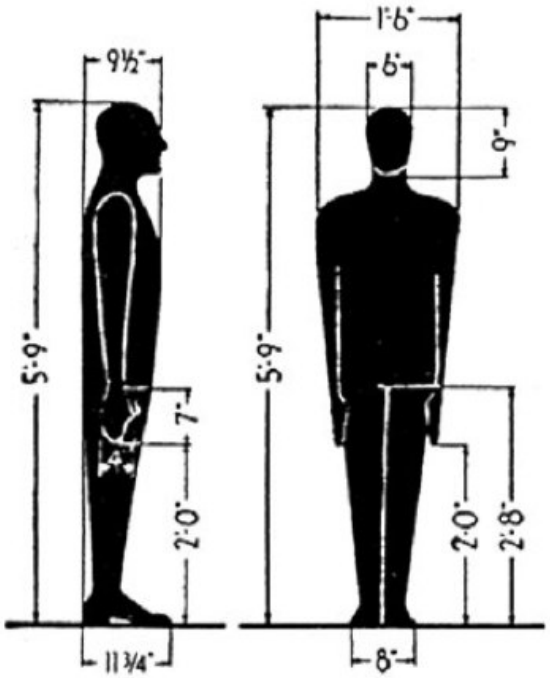
Hybridní metody využívají ke své detekci kombinaci různých postupů, ať již z kategorie učících se metod nebo různých matematických postupů. Představitele takovéhoto postupu můžeme najít v této práci [8]. K detekci hlavy a pozicí ramen je využíváno algoritmu AdaBoost. Pozice rukou jsou pak detekovány segmentací kůže. Hloubkový obraz je pak zpracován rozšířenou transformací vzdálenosti. Metoda sama o sobě příliš nezvládá detekci v reálném čase, ale autor nastiňuje možnosti její optimalizace.

Průběh algoritmu je rozdělen do několika částí. Nejdříve je pomoci hloubkového obrazu odstraněno pozadí, aby byl případem dalších studií pouze samotný objekt osoby. Na detekování hlavy a horní částí těla je použita metoda Viola-Jones vylepšená o AdaBoost. Nejdříve je detekovaná horní část těla a tato oblast se posílá do dalšího klasifikátoru, který má za úkol nalézt další oblast s obličejem.

Kostra je reprezentována sedmi klouby. Je použito antropometrických dat k určení velikostí částí lidského těla. Údaje k antropometrickým datům jsou čerpány z konkrétní knihy NASA, příklad lze spatřit na obrázku 1.7. Pozice kloubů hlavy je určena jako těžiště a střed spodní části oblasti, která byla výsledkem detekce obličeje. Pozice kloubů ramen je určena v půlce oblasti mezi čtvercem oblasti tváře a oblasti horní části těla. Šířka ramen je určena, jako dvojnásobek šířky obličeje. Délka paží je 1.12x násobek šířky tváří. Dále je vyhledávána pozice rukou a to pomocí segmentace kůže. K tomuto účelu je použit barevný model HSV, pomocí kterého je vytvořen binární obraz, který obsahuje oblasti kůže. V této fázi je již vypočítaná pozice kloubů hlavy, krku a ramen. Následuje proces vygenerování kostry rukou. V tomto případě je použita rozšířená transformace vzdálenosti aplikovaná na binární obraz obsahující postavu. Proces generování začíná v kloubu jednoho ramen a vygeneruje sadu čar o délce definované pomocí antropometrických dat. Snahou je čáru zapasovat do správné pozice v obrázku postavy a to pomocí hodnot z rozšířené transformace vzdálenosti. K určení pozice zápěstí jsou použity i údaje ze segmentace kůže.

Výsledný algoritmus dokáže s docela dobrou přesností generovat kostru postav, ale nezvládá některé hraniční případy jakými jsou překřížené ruce apod. Samotná rychlost algoritmu také není příliš závratná a největší podíl na tom má detekce oblastí pomoci AdaBoost.

Autor navrhuje několik vylepšení. Jedním je výměna AdaBoost metody pro detekci obličeje a horní části těla za metodu Template matching. Pro vylepšení přesnosti je navrhováno vytvoření popsané množiny dat obsahující kostry pro určité hloubkové obrazy.



Obrázek 1.6: *Antropometrické data[8]*

Základní myšlenka v podobě v podobě vzorkování přímek o dané vzdálenosti a hledání její schody s houbkovým obrazem je dále použita i v naší práci, v kapitole o návrhu systému je tak uvedeno mnohem rozsáhlejší a názornější osvětlení podobného principu.

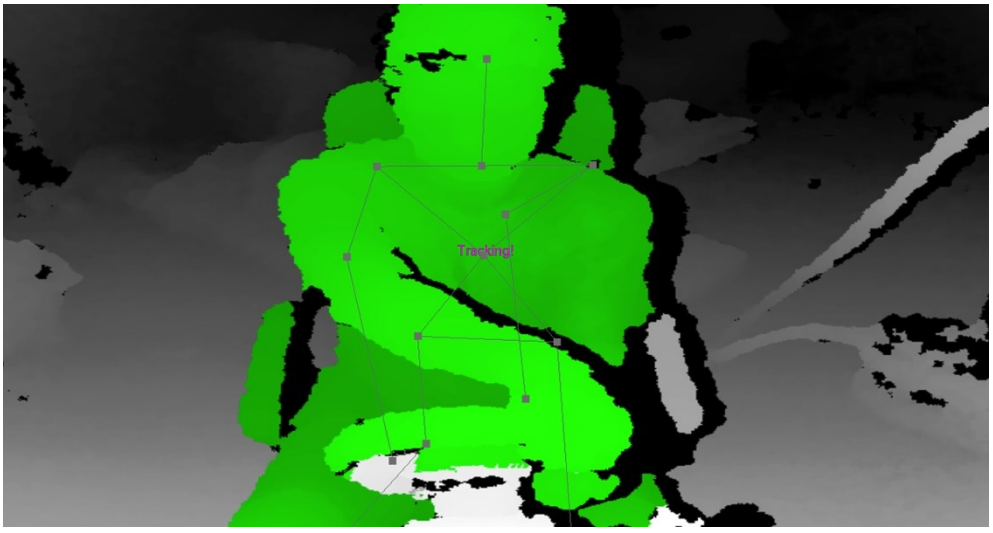
4 Nástroje pro detekci kostry

Tato kapitola se věnuje třem nejznámějším a volně dostupným knihovnám pro zařízení Kinect, které buďto v sobě nebo díky nějaké nadstavbě umožňují detekovat kostru postavy. Mezi nejznámější avšak dosti uzavřené patří SDK společnosti Microsoft, které se stalo oficiální vývojovou platformou pro zařízení této společnosti. Za zmínku určitě stojí také OpenNI, které nabízí platformu pro široké spektrum zařízení a jehož snaha je vytvořit rozhraní pro přirozenou komunikaci uživatele s počítačem. Trojici uzavírá knihovna Open Kinect. Jedná se o volně dostupnou knihovnu, která položila základy vývoje pro zařízení Kinect a z celé trojice je nejotevřenější.

4.1 OpenNI

Dle [9] se jedná o neziskovou organizaci, která si klade za cíl sjednotit přístup k zařízením přirozené interakce. Součástí je sada knihoven OpenNI SDK, která umožňuje komunikovat s řadou zařízení poskytujících hloubkový obraz. Sama o sobě je OpenNI jenom vrstvou umožňující snadnou komunikaci s různými zařízeními přirozené interakce, jakými jsou například Kinect nebo Asus Xtion. Proto je v architektuře nástroje vytvořena ještě vrstva middleware, do které je umožněno přispívat různým členům OpenNI komunity a obsahuje nástroje pro detekci gest, kostry nebo 3D rekonstrukci, které využívají rozhraní OpenNI.

Jedním s takových nástrojů je i NiTE, který umožňuje sledovaní dlaně a detekci kostry. Algoritmus, který byl použit pro detekci nebyl popsán, existuje však dokument[10], který obsahuje určitá doporučení a omezení algoritmu. Jednou s vlastností, která je pro nás dosti omezující je nutnost kalibrace stojící osoby. NiTE potřebuje osobu nejdříve vysegmentovat a provést kalibraci. Také je uvedeno, že algoritmus podává nepřesné výsledky při vzdálenosti menší, než 2,5 metru a pokud se v okolí osoby nacházejí nějaké předměty. Problém s okolními předměty by v našem případě mohl být vyřešen odstraněním statického pozadí, avšak NiTE od verze 2 znemožnil upravovat vstupní hloubkový obraz.

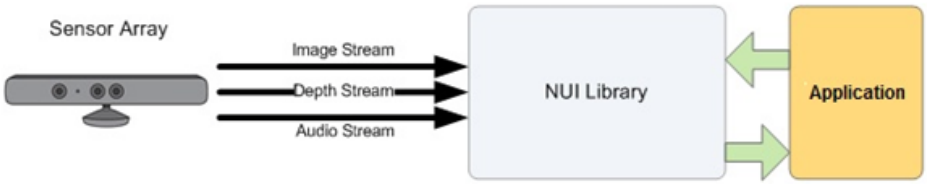


Obrázek 1.7: *Detekce v OpenNI*

Na obrázku výše lze vidět chování NiTE v improvizovaných podmínkách simulující prostředí oblasti řidiče. Jako problém lze vidět, že algoritmus zahrnul do oblasti postavy i části židle a volantu. Detekce je tak i možná z tohoto důvodu značně chybová. Na odhadnuté pozici rukou lze vidět, že knihovna počítá i s volantem ve svých odhadech.

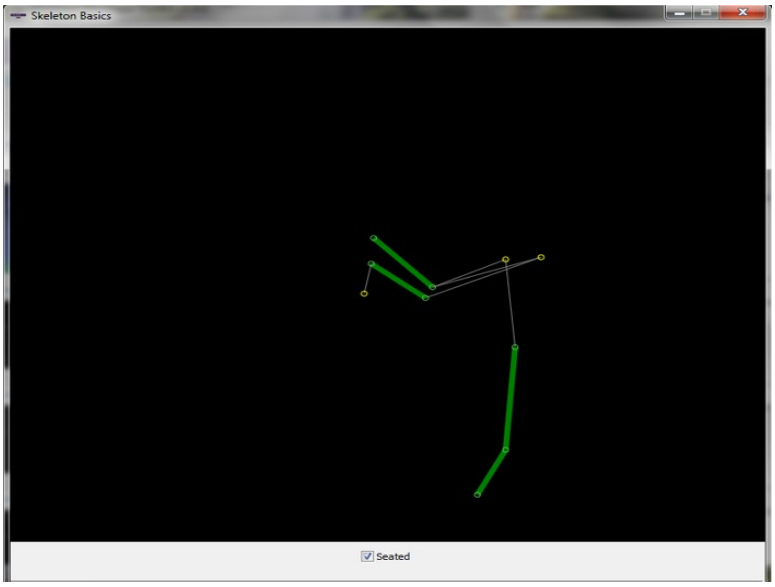
4.2 Microsoft SDK

Jedná se o nástroje primárně určené pro zařízení Kinect, vytvořené společností Microsoft, umožňující vytvářet aplikace pro Kinect v jazycích C++, C#, Visual Basic a určené pro systémy Windows. Součástí architektury, která je zobrazena na obrázku 1.9, jsou tři hlavní komponenty. První tvoří pole senzorů Kinect. Druhou je NUI knihovna poskytující aplikaci nástroje pro přirozené uživatelské rozhraní (detekce kostry, gest, apod.). Tato knihovna komunikuje přímo s ovladači pro Kinect. Třetí komponentou je naše samotná aplikace využívající NUI knihovnu. Možnou výhodou je možnost práce s více zařízeními Kinect, což umožňuje dopočítat hodnoty, které jsou jinému zařízení skryty.



Obrázek 1.8: *Architektura Microsoft SDK[11]*

Samotná detekce kostry je založena na algoritmu publikovaného v [7]. Na obrázku níže lze vidět, že aplikace trpí mnohem většími problémy, než výše zmíněná NiTE knihovna. Téměř při každém dotyku osoby s volantem došlo k značnému rozhození kostry. A je také problémem, že je knihovna značně uzavřená a nedovoluje upravit data vstupující do algoritmu detekce kostry.

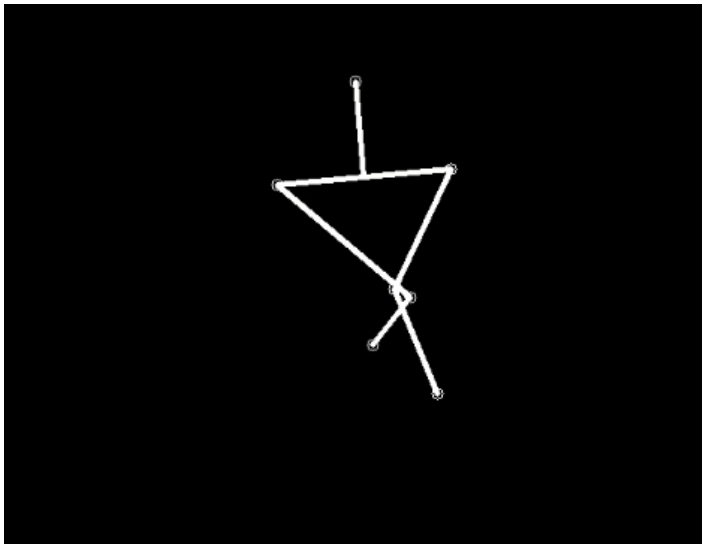


Obrázek 1.9: *Detekce v Microsoft SDK*

4.3 Open Kinect, Skeltrack

OpenKinect je otevřenou knihovnou pro zařízení Kinect, která poskytuje přístup k hloubkovému a barevnému obrazu z tohoto zařízení. Jedná se o jednu z prvních knihoven, které umožnili vývoj aplikací na tomto zařízení. Sama o sobě neposkytuje žádné algoritmy pro detekci kostry nebo gest, ale díky otevřené komunitě již existují některé projekty pracující nad touto knihovnou.

Jedním z takových projektů je Skeltrack. Jedná se o knihovnu implementující postupy uvedené v [12] pro detekci kostry horní poloviny těla. Knihovnu uveřejnil Joaquim Rocha. Na rozdíl od této práce však autor uvádí, že nebyla využita databáze póz, ale místo toho použity matematické postupy a heuristika. Jednou z výhod je široká možnost ovlivňování vstupů pro tuto knihovnu a jednoduchá konverze výstupů do dalších knihoven, jako například OpenCV. Knihovna nepotřebuje pro detekci žádnou kalibraci a detekce funguje ihned po spuštění. Výsledky jsou opět velice chybové při doteku volantu oběma rukama.



Obrázek 1.10: *Detekce v knihovně Skeltrack*

Výstup detekce knihovny Skeltrack lze znovu spatřit na obrázku výše. Výsledky jsou podobně nekvalitní, jako v knihovně od společnosti Microsoft. Je však očekáváno, že výsledky mohou být dobře ovlivnitelné úpravou vstupujícího hloubkového obrazu nebo přenastavením parametrů detekce.

5 Návrh systému

Tato kapitola se věnuje návrhu systému pro detekci kostry postavy v RGBD obraze a popisu jeho procesů. V první části se čtenář seznámí se základní charakteristikou řešeného problému a je obeznámen s důvody, jež vedly k zvolení konkrétních řešení. V dalších kapitolách jsou vybrány použité technologie pro implementaci výsledných postupů a jsou blíže rozebrány konkrétní řešení. Konkrétně byly navrženy dvě metody pro detekci, první využívá k detekci loktů a dlaní transformaci vzdálenosti a druhá se snaží o hledání dlaní v okolí volantu pomocí segmentace barvy kůže. V závěru této kapitoly je také navržena sada gest a jejich detekce. Tyto gesta mohou být použity například pro ovládání palubních systémů nebo navigace automobilu.

5.1 Popis problému

Je potřeba vytvořit systém, který bude schopen generovat kostru postavy řidiče sedícího za volantem uvnitř vozidla použitím zařízení Kinect. Na rozdíl od dosud zveřejněných systémů detekce kostry, většinou navržené pro použití v oblasti herního průmyslu, se v naší aplikaci setkáváme s několika překážkami, které komplikují použití dostupných aplikací.

Jednou s prvních komplikací je použití v kokpitu vozidla. Je obtížné nalezení správné pozice zařízení Kinect, aby byla správně snímána celá oblast řidiče. Zařízení také podává o něco horší výsledky hloubkového obrazu v otevřeném prostoru automobilu, na rozdíl od místnosti. Mezi hlavní komplikace patří množství nadbytečných objektů, které se nacházejí v oblasti řidiče. Můžeme jmenovat například volant, bezpečnostní pásy nebo křeslo. Všechny tyto problému způsobují, že dosud existující aplikace selhávají. Mnoho jich je optimalizováno pro analyzování stojící postavy. Pokud již zvládne i postavu sedící, nastává problém, pokud řidič uchopí volant.

Řešení vybízí ke dvěma možnostem. První možností je kompletní re-implementace jedné z metod, kde by se pro natrénování použila nová sada postav sedících za volantem. Druhou možností je použít některou z dostupných aplikací a navrhnout systém korekce získaných výsledků. Tato práce se dále bude věnovat metodě druhé a navrhne dva možné algoritmy. Jeden komplexnější detekující pozice loktů a dlaní a druhý jednodušší, detekující pouze dlaně v okolí volantu. Jelikož tato metoda počítá s nadstavbou algoritmů nad již existujícím řešením, je potřeba aby tyto algoritmy byly co nejefektnější a zbytečně neznemožňovaly použití aplikace zbytečnou degradací výkonu.

5.2 Výběr použitých technologií

Vzhledem k tomu, že se vydáváme cestou značné úpravy a zásahu do výpočtu, je třeba zvolit systém, který je dostatečně otevřený. Je potřeba, aby nám tento systém povolil upravit hloubkový a barevný obraz, který vstupuje do detekčního algoritmu a také aby nám dal dostatek informací, které bychom pak mohli využít v dalších knihovnách pro zpracování obrazu. Jako systém pro detekci kostry byla vybrána knihovna Skeltrack. Jedná se o volně dostupnou a dobře

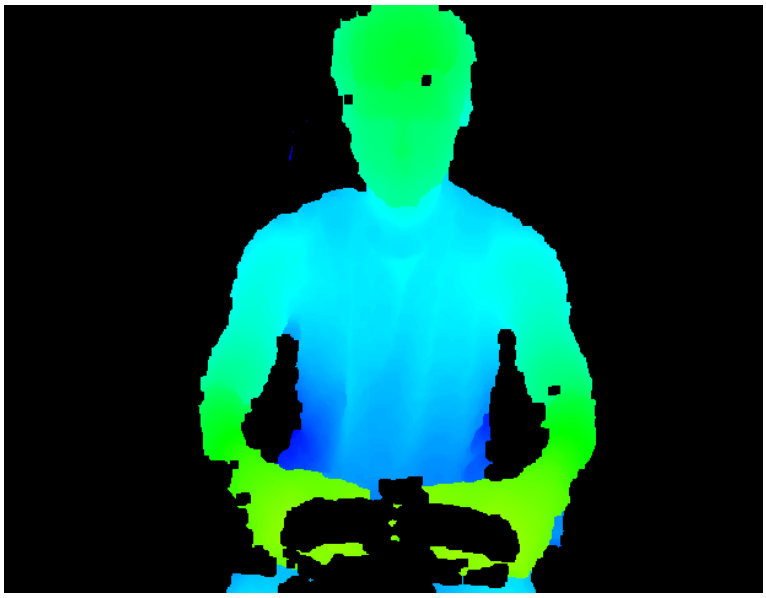
škálovatelnou knihovnu pro detekci kostry, kde máme možnost i měnit zpracovávaná data, což je v našem případě velmi důležité. Knihovna také nabízí spoustu parametrů k nastavování, což vybízí k experimentování a možnosti dosažení lepších výsledků. Autor také uvádí zdroj naimplementované metody, která je popsaná zde [12]. Nejsou však použita trénovací data.

Jelikož nám Skeltrack nebude stačit, potřebujeme vybrat ještě knihovnu pro úpravu a práci s obrazem. Pro aplikaci různých již dobře známých algoritmů a postupů zpracování obrazu byla vybrána knihovna OpenCV. Jedná se o multiplatformní knihovnu, která je již v této oblasti velice dobře známá a široce používaná. Umožňuje také vytváření jednoduchého uživatelského rozhraní.

5.3 Odstranění pozadí a statických objektů

Vzhledem k tomu, že v prostoru řidiče se nachází předměty se kterými většina algoritmů nepočítá, je potřeba je ještě před vstupem do algoritmu nějak smysluplně odstranit. Jelikož pozice kamery bude statická, nabízí se v tomto případě vytvoření modelu pozadí. Model pozadí bude vytvářen z hloubkového obrazu. Při zpracování pak máme informaci v kterých hloubkách se nacházejí body, které nepotřebujeme, ty následně v dalších zpracováních vypouštíme.

Na obrázku 1.12 vidíme takto upravený hloubkový obraz. Můžeme spatřit, že bylo odstraněno křeslo a volant. Díky odstranění volantu je oblast kolem volantu částečně deformovaná a může docházet k částečné ztrátě některých informací. Takto upravený hloubkový obraz bude vstupem pro oba algoritmy detekce.



Obrázek 1.11: *Výsledek odstranění pozadí v hloubkovém obraze*

5.4 Kalibrace barevného a hloubkového obrazu

Jelikož je pozice infračerveného a barevného snímače odlišná dochází výsledcích ke zkreslení v pozicích bodů mezi těmito obrazy. Pozice bodu v barevném obraze není stejná jako ta v hloubkovém, proto je třeba provést kalibraci, která přepočítá pozici v hloubkovém obrazu na pozici v barevném. Jinak bychom nemohli například pomocí hloubkového modelu pozadí správně odstraňovat pozadí i v barevném obraze. Používají se k tomu klasické stereo kalibrační techniky a postup k takovému výpočtu můžeme vidět níže.

P3D.x = (x_d - cx_d) * depth(x_d,y_d) / fx_d

P3D.y = (y_d - cy_d) * depth(x_d,y_d) / fy_d

P3D.z = depth(x_d,y_d)

P3D' = R.P3D + T

P2D_rgb.x = (P3D'.x * fx_rgb / P3D'.z) + cx_rgb

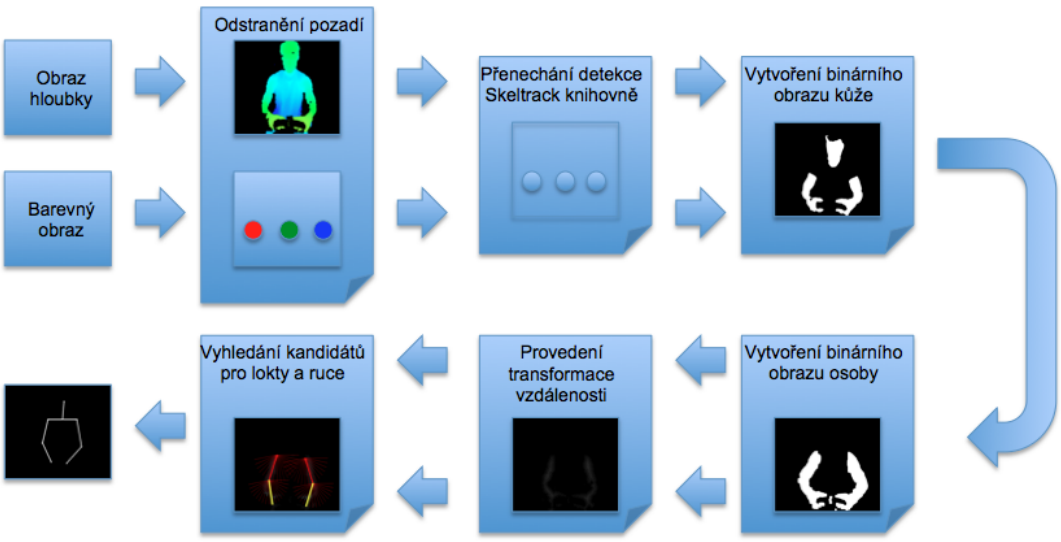
P2D_rgb.y = (P3D'.y * fy_rgb / P3D'.z) + cy_rgb

x_d, y_d jsou souřadnice v hloubkovém obraze, depth(x_d,y_d) je funkce vracející hloubku v daném bodě

R a T jsou matice pro rotaci a posun vyměřené kalibrací, ostatní proměnné určují další zkreslení a byli pro ně použity již změřené hodnoty dostupné zde [16]

5.5 Metoda hledání loktů a dlaní pomocí transformace vzdálenosti

Tato metoda je inspirována z upraveného postupu uvedeného v [8]. Jelikož knihovna Skeltrack odvádí dobrou práci širším okolí volantu. Je tato metoda odpovědna pouze na hledání dlaní a loktů, jen v případě, že se dlaně nacházejí v definované krizové oblasti kolem volantu. Tímto způsobem můžeme docílit relativně dobré úspěšnosti detekce. Celý proces je zjednodušeně znázorněn na následujícím obrázku.



Obrázek 1.12: *Postup hledání loktů a dlaní*

Do algoritmu vstupuje hloubkový a barevný obraz. Hloubkový obraz je přenechán knihovně Skeltrack. Výstupy této knihovny jsou použity pro určení pozice ramen a hlavy. Následně je z hloubkového obrazu vytvořen binární obraz osoby a z barevného binární obraz kůže. Na binární obraz postavy je aplikována transformace vzdálenosti a takto upravený obraz je prohledáván sadou parametrizovaných přímek pro určení pozic loktů a dlaní. Mnohem detailněji jsou všechny tyto procesy popsány v následujících kapitolách.

5.5.1 **Vytvoření binárního obrazu kůže**

Bylo usouzeno, že je možné některé nedostatky způsobené odstraněním pozadí v hloubkovém obraze vylepšit informací o pozici kůže v obraze. Proto je vytvořen binární obraz, jehož body mají pozitivní hodnotu v pozicích, kde se nachází kůže. Pro tento účel byl využit barevný model YCrCb, jehož rozsahy pro hodnotu kůže jsou dány tabulkou.

	MIN	MAX
Y	0	255
Cr	133	173
Cb	77	127

Tabulka práhových hodnot modelu YCrBr pro kůži

Jelikož máme díky knihovně Skeltrack prostorovou pozici ramen, můžeme si také dovolit prohledávat pixely, které se nachází v prostoru před rameny. Výsledek takovéhoto zpracování můžeme vidět na obrázku 1.14.

Obrázek 1.13: *Binární obraz kůže postavy*

5.5.2 Vytvoření binárního obrazu postavy

Díky odstranění pozadí z hloubkového obrazu by se zde již měli nalézat body náležející pouze postavě. Je proto možné označit tyto body, jako body postavy a vytvořit z nich binární obraz. Zde jsme navrhli také mírné vylepšení a díky tomu, že máme k dispozici pozice kloubů, pokoušíme se z tohoto obrazu odstranit trup. Postava je v této fázi rozdělena na dvě poloviny. Tyto poloviny rozděluje střed mezi levým a pravým ramenem a ukončuje pozice jednotlivých ramen. Body z každé poloviny jsou vynechány, pokud jejich prostorová pozice je větší (hlubší), než pozice konkrétního ramene. Výsledek můžeme vidět na obrázku 1.15.

Jak je vidět, díky tomuto zpracování máme k dispozici vysegmentované paže a ulehčeno tak další zpracování nebo prohledávání prostoru.

Obrázek 1.14: *Binární obraz postavy bez torza*

5.5.3 Aplikace transformace vzdálenosti

[13]Výsledkem takovéhoto procesu je takzvaná mapa vzdálenosti, kdy každý pixel nese informaci o vzdálenosti k nejbližší překážce. Tato operace se užívá na binárních obrazech, překážkou je v tomto případě myšlena hranice konkrétní oblasti. Tato operace se velice často užívá ke skeletonizaci binárních obrazů, kde se pak hledají maxima a dochází k maximálnímu ztenčení objektů. Princip takové transformace můžeme vidět na obrázku 1.6. Tato transformace je zde provedena na binárním obraze obdélníku.

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

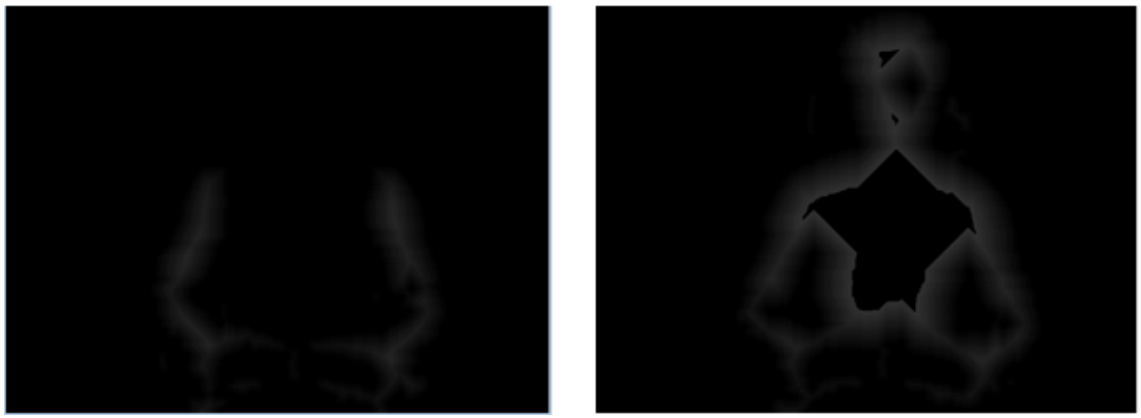
Binary Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	2	2	2	1	0
0	1	2	3	2	1	0
0	1	2	2	2	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

Distance transformation

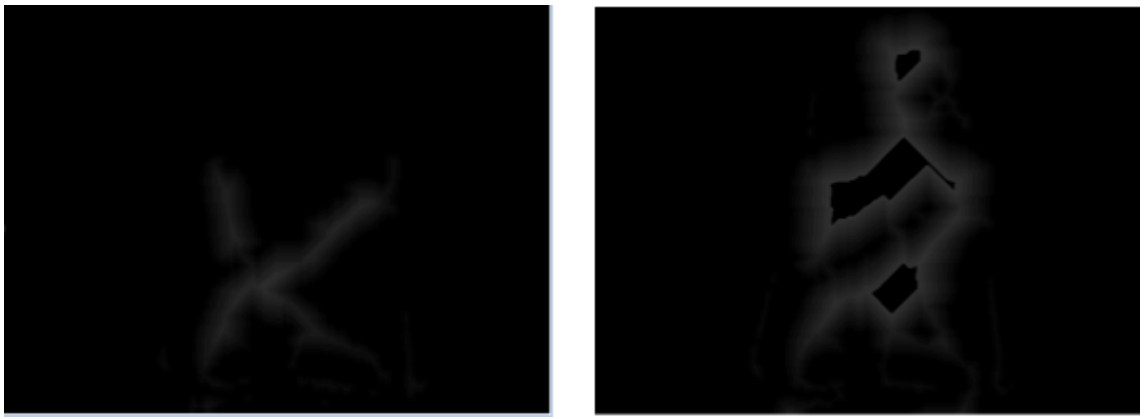
Obrázek 1.15: *Hodnoty transformace vzdálenosti na binárním obraze[13]*

Napravo jde již vidět obraz kde máme údaje o vzdálenostech k hranicím. V našem případě jsme tuto transformaci provedli na náš binární obraz postavy a výsledek lze spatřit na obrázku níže.



Obrázek 1.16: *Aplikace transformace vzdálenosti na binární obraz postavy*

Jsou uvedeny pro srovnání dva příklady. Na levém obrázku je transformace aplikována na obraz s odstraněným trupem a hlavou a napravo bez této úpravy. Jak je vidět, tato úprava nám výrazně ulehčuje prostor prohledávání. Tento rozdíl je ještě markantnější v příkladu níže, kdy je transformace aplikována na binární obraz postavy s překříženýma rukama. Bez odstranění trupu by hledání správné pozice loktů bylo jen velice obtížné.

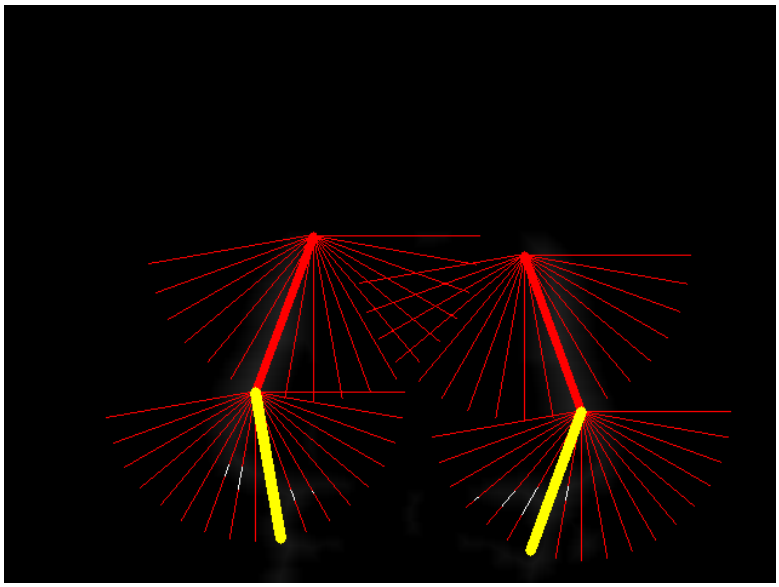


Obrázek 1.17: *Aplikace transformace vzdálenosti na binární obraz postavy s překříženýma rukama*

5.5.4 **Vyhledání kandidátů pro lokty a dlaně**

Nyní jsme již v poslední fázi algoritmu. Tato část má za úkol nají v obrazu z předchozí transformace kandidáty na pozice dlaní a loktů a najít ten nejvhodnější. Nejdříve je tedy hledán kandidát pro loket. Je vycházeno z pozice ramen, které máme k dispozici díky knihovně Skeltrack. Z této pozice je vzorkována sada přímek pod daným úhlem a délkou danou antropometrickými daty. Dle těchto dat je délka mezi loktem a ramenem určena jako násobek

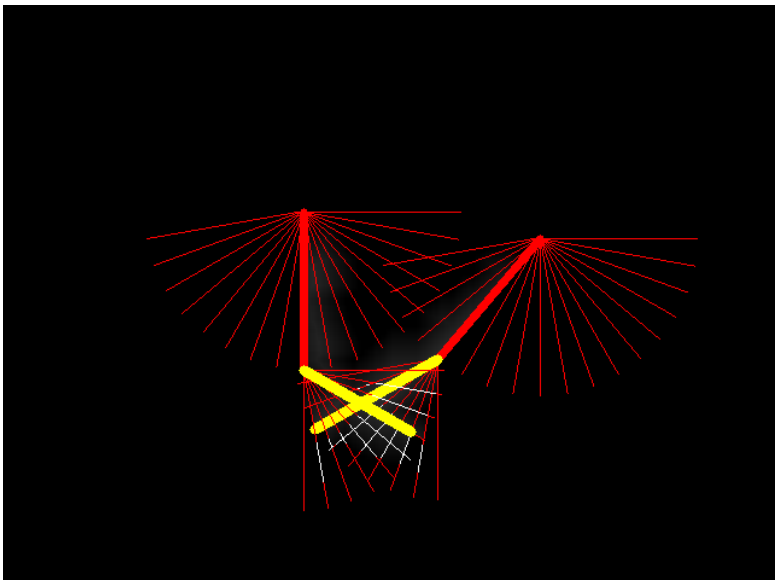
šířky hlavy. V mém případě používám násobek 1.12. Šířka hlavy je zpracována z hloubkového obrazu pomocí její pozice vyplněním této oblasti a zjištěním její šířky. Celý proces je znázorněn na obrázku níže.



Obrázek 1.18: *Vyhledávání kandidátů v obraze po aplikaci transformace vzdálenosti*

Z těchto přímek jsou vybráni nejsilnější kandidáti, kteří protínají body nesoucí největší hodnoty po aplikaci transformace vzdálenosti. Z těchto kandidátů na lokty se pak podobným způsobem prohledává prostor, aby se nalezli pozice rukou. Konec přímky je pak dán posledním pixelem pozice, ve které se ještě nachází kůže. Pokud nelze nalézt dlaň z dané pozice loktu, ve které by se nacházelo přijatelné množství kůže, pokračuje se dalším kandidátem na loket. Vybraní kandidáti jsou pak znázorněni tlustými čarami. Jelikož barevná informace o kůži nemusí být úplně stabilní pro různé typy lidí a při změnách osvětlení, je navržena i možnost prohledávání prostoru předloktí bez informace o kůži. V tomto případě se v obraze na němž je aplikovaná transformace vzdálenosti hledá přímka, která protíná největší hodnoty vzdálenosti a pokrývá nejvíce nenulových hodnot. Pozice dlaní je pak určena délkou takto nenulových hodnot. Mezi těmito dvěma metodami pro určení dlaní by měla být možnost ve výsledné aplikaci přepínat a může to být možností závěrečných porovnávání.

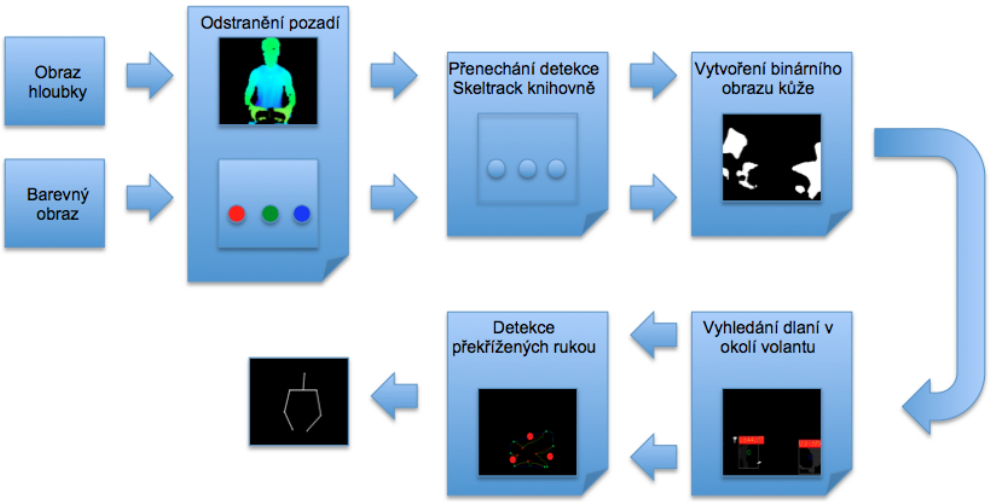
Na dalším obrázku je ještě znázorněna situace, kdy má osoba překřížené ruce. V tomto případě je vidět úprava, která způsobuje, že pokud úhel loktu směřuje dovnitř torza osoby, vzorkování kandidátů na ruce je omezeno pouze na poloviční interval.



Obrázek 1.19: *Vyhledání kandidátů při překřížení rukou*

5.6 Metoda hledání dlaní v okolí volantu

Vedle předchozí komplexní metody je navržena ještě jedna samostatná metoda, která je navržena na poněkud triviálnějším základu a jejím hlavním cílem je podání co největšího výkonu v detekci. Tato metoda funguje samostatně vedle předchozí a je určena pro závěrečné srovnání.

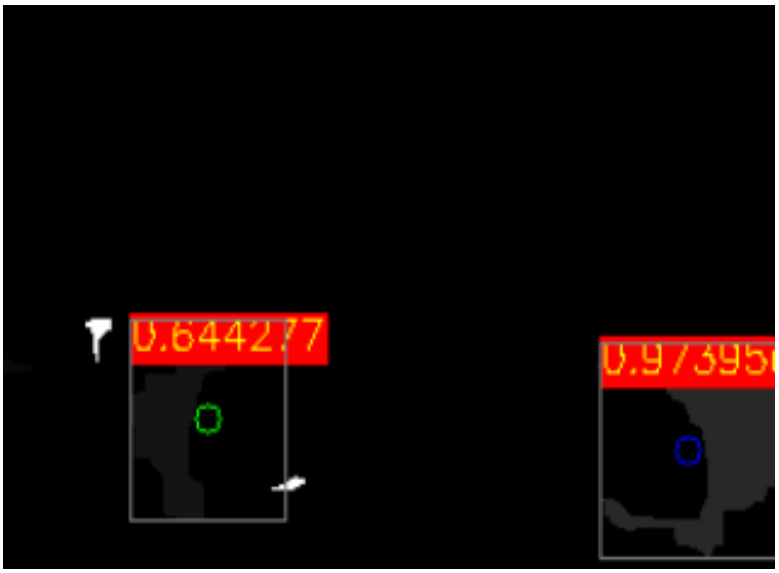


Obrázek 1.20: *Postup detekce dlaní v okolí volantu*

Zjednodušené schéma částí algoritmu lze opět nalézt na obrázku výše. Základ v podobě odstranění pozadí a detekce knihovny Skeltrack je podobný s předchozím postupem. Na rozdíl od předchozího případu však využíváme i pozice loktů. Prací našeho algoritmu je hledání dlaní v oblasti kolem volantu. Oblast prohledávání je omezena pouze na tuto oblast, pro kterou máme

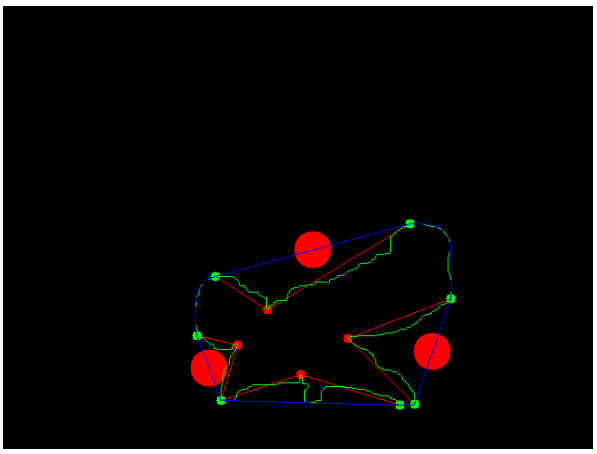
také k dispozici hloubkový obraz s odstraněným pozadím a trupem a také binární obraz kůže. Tato oblast je prohledávána a jsou segmentovány objekty, které pokrývá kůže s určitým minimálním pokrytím. Oblast musí mít také svou minimální velikost a také musí splňovat určité parametry podlouhlosti. Pokud se v této oblasti nalézají dva objekty splňující tyto kritéria, jsou určeny jako dlaně. Jako levá dlaň je určena ta, která se nachází na levé straně a jako pravá ta na pravé straně. Ukázku lze vidět na obrázku níže.

Čísla v rámečku označují procentuální pokrytí objektu kůží. Segmentované ruce jsou označeny různými stupni šedé.



Obrázek 1.21: *Nalezené oblasti dlaní*

Problém však nastává v případě, že má osoba překřížené ruce. V tomto případě je určení pozic dlaní převrácené. Metoda proto dále pokračuje detekcí překřížení rukou. Ta je prováděna nad binárním obrazem osoby bez trupu. Je vytvořen konvexní obal nad rukama, které jsou považovány za zkřížené a jsou vyhledány případy, které porušují konvexnost. Pokud jsou nalezené tyto případy tři o určité minimální velikosti, je považováno, že má osoba zkřížené ruce. Příklad takovéto detekce můžeme vidět na obrázku níže. Pozice levé dlaně je pak s pravou vyměněna.



Obrázek 1.22: *Detekce překřížených rukou*

5.7 Detekce gest

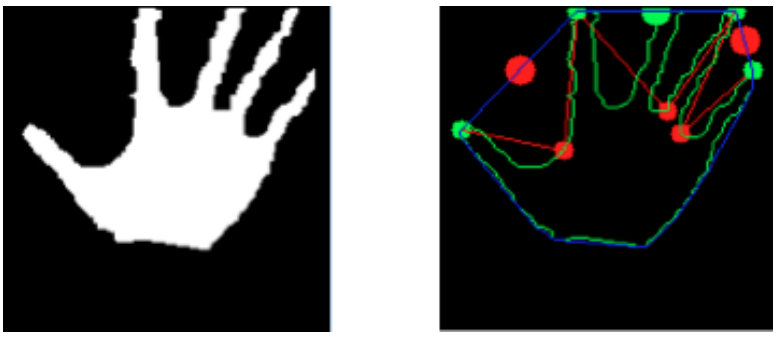
Jelikož by mohla mít aplikace specifické využití v oblasti komunikace mezi řidičem a systémy automobilu je součásti návrhu také zamyšlení nad možným využitím a návrh základní sady gest, které by pomocí našeho systému šlo identifikovat.

Vydal jsem se cestou návrhu gest, které by mohly být využity pro komunikaci s palubním počítačem automobilu nebo také navigací. Systém v základní verzi by měl umět detekovat čtyři základní typy gest. Jsou jimi události, kdy řidič položí obě ruce na volant. Událost, kdy je pravá ruka zdvižená a levá na volantu. Tato událost by měla sloužit pro přepnutí systému do režimu ovládání. Uživatel touto pravou rukou pohybovat a ovládat tak pozici kurzoru v cílovém systému. Gesto položení rukou na volant může tento režim vypnout. Poslední dva gesta by měla sloužit k vyvolání akce v systému. Jedná se o událost kdy uživatel otevře dlaň a událost kdy uživatel dlaň zavře. Pomocí těchto gest lze například simulovat kliknutí myší, kdy uživatel v sekvenci a určitém časovém intervalu zavře a otevře dlaň. Rozlišení mezi otevřenou a zavřenou dlaní může také rozlišovat různé akce, které vykonává pohybující se dlaň. Například zavřená dlaň může sloužit k uchopení nějakého objektu (mapy) a posouvání, případně přibližování, díky tomu, že máme prostorovou informaci o pozici dlaně.

Gesta pro položení rukou na volant a zvednutí pravé ruky, lze v našem systému detekovat snadno, protože máme anotovanou oblast volantu. Gesta otevření a uzavření dlaně budou osvětlena v následující kapitole.

5.7.1 Detekce otevřené a zavřené dlaně

Pro rozlišení mezi otevřenou a zavřenou dlaní používáme námi již dobře známý binární obraz vytvořený z hloubkového obrazu s odstraněným pozadím. Díky naší generované kostře máme informaci o prostorovém umístění dlaně. Tuto dlaň si tak do určité úrovně hloubky vysegmentujeme. Segmentuje se v okolí hloubky danou hloubkovou pozicí dlaně. Jak je vidět na obrázku níže takto vytvořený binární obraz je relativně dobře čitelný a nejspíš také odolnější proti chybám, než některé metody založené na segmentaci kůže.

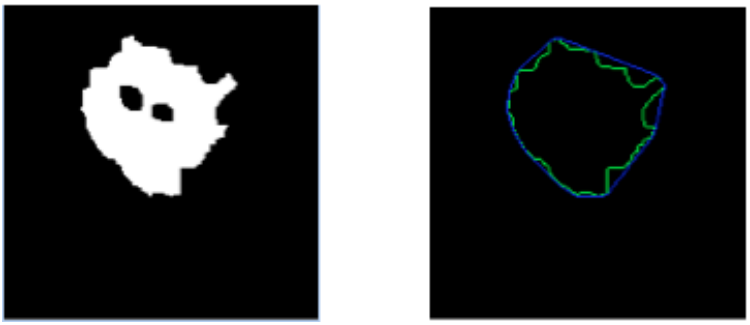


Obrázek 1.23: *Detekce otevřené dlaně*

Nad takovýmto binárním obrazem je provedeno hledání kontur a vytvořen konvexní obal, který lze na pravém obrázku vidět označen modře a v původním objektu jsou hledány poruchy této

konvexnosti. Výsledek je vidět také v obrázku napravo. Pokud je metoda úspěšná, lze velice snadno i určit kolik prstů dlaň ukazuje. V našem případě se však spokojíme, pokud se v objektu bude nalézat alespoň jeden takto charakteristický defekt. Uživatel totiž může prsty příliš blízko u sebe a výrazná pak bude pouze mezera mezi palcem a ukazováčkem.

Na posledních dvou obrázcích lze vidět moment, kdy uživatel uzavře dlaň. V této chvíli v objektu nejsou žádné výrazné konvexní defekty a dlaň je klasifikována, jako uzavřená.



Obrázek 1.24: *Detekce zavřené dlaně*

6 Implementace

V této kapitole je rozebrána praktická stránka této práce. Cílem totiž nebylo jen dané řešení navrhnout, ale také naimplementovat a prozkoumat možnosti užití. Čtenář se v této kapitole dozví, jak výsledná aplikace vypadá, jak byla navržen její kód a také možnosti její konfigurace a použití. Samozřejmostí jsou také informace o použitých pomocných knihovnách.

6.1 Technická specifikace

Aplikace byla implementována v operačním systému Ubuntu Linux 13.10 a ve vývojovém prostředí Eclipse Ganymade nakonfigurovaném pro vývoj v jazyce C++. Využita byla knihovna Skeltrack ve verzi 0.1.14 a OpenCV 2.4.8.

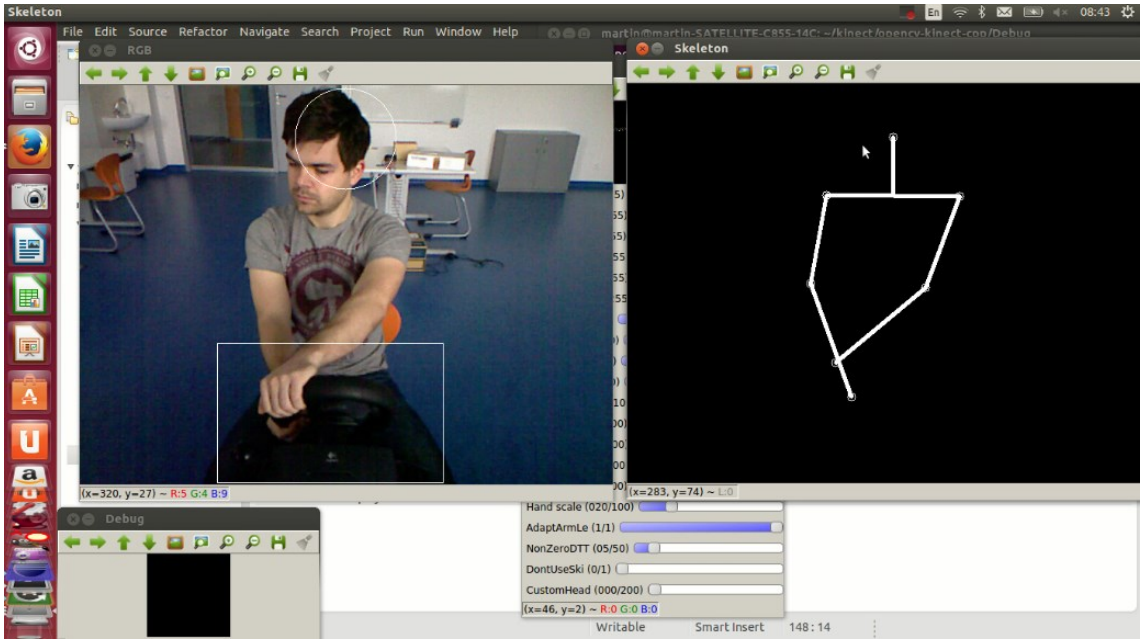
6.2 Popis uživatelského prostředí

Ve vytvořeném uživatelském prostředí bylo cíleno na jednoduchost, neboť to je ve výsledku určeno k dostatečnému otestování algoritmů. Tvoří jej hlavní okno s generovanou kostrou, okno přenášející barevný obraz, ve kterém lze zhlédnout anotovanou oblast kolem volantu, vyznačenou obdélníkem a také kružnici podávající informaci o naměřené šířce obličeje. Součástí aplikace je také okno umožňující měnit za běhu výsledky algoritmů, nastavováním jejich parametrů pomocí posuvníků.

Oblast kolem volantu lze přenastavit označením myší levého horní rohu a kliknutím a označení stejným způsobem pravého dolního rohu. Obdélník označující oblast se ihned po nastavení pravého dolního rohu aktualizuje. Aplikace také umožňuje ovládání několika funkcí klávesnicí. Tabulku zobrazující přehled o použitých klávesách můžeme vidět níže.

Klávesa	**Popis funkcionality**
t	Slouží ke spuštění trénování modelu pozadí. Ve zpuštění je 10s zpoždění, aby si uživatel stihl poodstoupit a nebyl v záběru, po dokončení je zobrazen hloubkový model pozadí a je možno pokračovat. Model je uložen na disk a načten při dalším spuštění programu.
1/2	Klávesa 1 slouží pro přepnutí detekce do režimu detekce metodou transformace vzdálenosti. Klávesa 2 přepíná do režimu detekce dlaní v okolí volantu.
a	Slouží k zobrazení binárního obrazu osoby.
s	Slouží k zobrazení binárního obrazu kůže.
d	Slouží k zobrazení obrazu po aplikaci transformace vzdálenosti a představuje taky částečně průběh detekce vykreslením vygenerovaných čar.

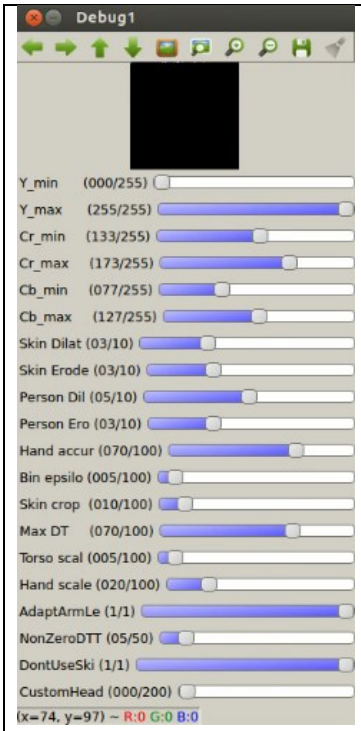
Dále můžeme vidět ukázku z výsledného programu.



Obrázek 1.25: *Vzhled aplikace*

Svůj popis si určitě zaslouží i okno ladění skryté na obrázku v pozadí. Změnou parametrů totiž můžeme značně ovlivnit výsledky.

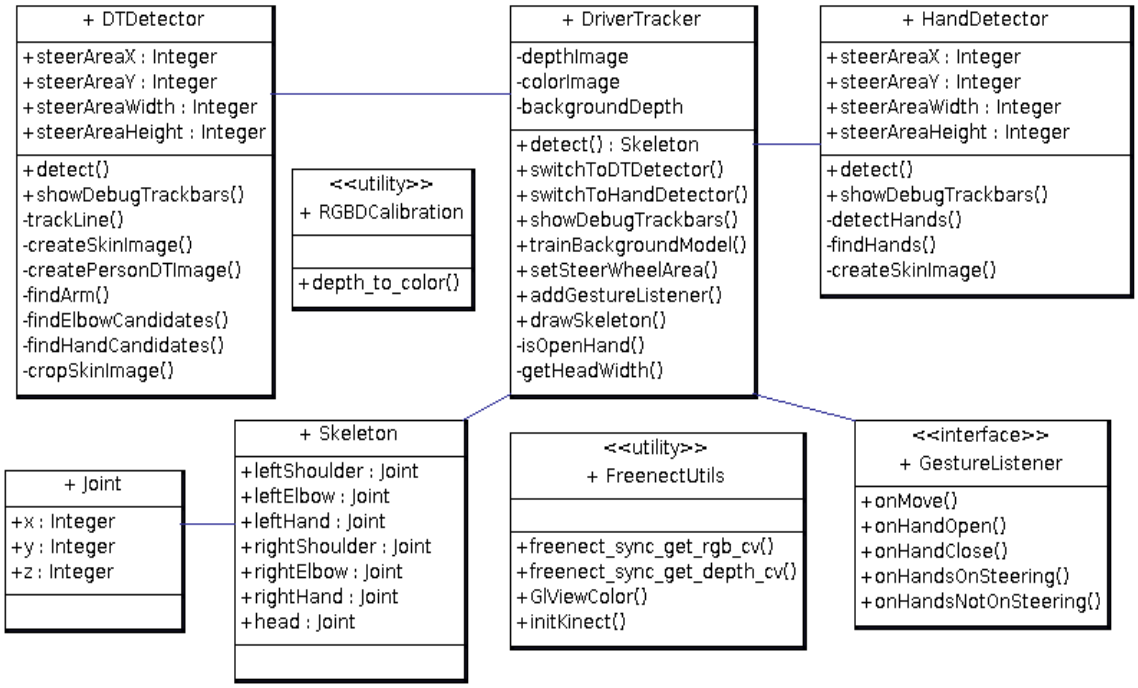
	Y,Cr,Cb (min,max) – slouží k nastavení prahů procesu pro detekci kůže.
	Skin dilate | erode – určuje míru dilatace a eroze pro binární obraz kůže.
	Person dilate | erode – určuje míru dilatace a eroze pro binární obraz osoby.
	Hand accur – určuje minimální přesnost v procentech pro detekci pozice dlaně
	Bin epsilon – jedná se o proměnnou určující, jak moc se bude ořezávat torzo osoby.
	Skin crop - proměnná určující jak moc do hloubky se bude ořezávat binární obraz kůže.
	Max DT – určuje maximum, které hodnoty transformace

	vzdálenosti se ještě budou počítat.
	NonZeroDTT – určuje minimální práh pro určování pozice kloubu z obrazu transformace vzdálenosti.
	Torso scale – určuje, jak moc do šířky se bude ořezávat torso osoby.
	Hand scale – určuje prodloužení hledané pozice dlaně o násobek v procentech
	AdaptArmLength – zapíná vlastnost adaptování délky vzorkované přímky pro pozici loktu.
	DontUseSkin – pokud je nastaven, tak metoda detekce nepoužívá informaci o kůži, ale používá pouze hloubkový obraz.
	CustomHead – možnost vypnout určování délky přímek ze šířky hlavy a nastavit šířku vlastní. Při experimentování lze tuto vlastnost nastavit na maximum spolu se zapnutou vlastností AdaptArmLength. Jedná se o dobrou alternativu určování délek vzorkovaných přímek.

Aplikace také podává informace o výstupu z detekce gest. Informace je podána na standardním výstupu příkazového řádku. Konkrétně jsou vypisovány tyto zprávy – „Hands on steering“, „Hands not on steering“, „Hand opened“, „Hand closed“ a „Hand clicked“. Gesto „Hand clicked“ označuje událost kdy uživatel zavřel a otevřel dlaň v intervalu menším než 2 sekundy.

6.3 Objektový návrh aplikace

Aplikace je naimplementována v jazyce C++ a snaží se o udržitelný objektový návrh. Jedním z důvodu je pozdější možná využitelnost a rozšiřitelnost. Každá oblast je zapouzdřena do speciálně pro tento účel navrhnuté třídy. Pro rychlý přehled o rozhraní tříd byl vytvořen třídní diagram, který můžete spatřit na obrázku níže. V diagramu jsou zahrnuty pouze hlavní třídy navržené knihovny a pro přehlednost neobsahuje parametry metod, protože celý diagram by tak neúměrně narostl na velikosti. Celkově je kód zamýšlen jako knihovna, která by mohla být prakticky využitelná v dalších aplikacích, které by pak našli konkrétní využití v oblasti detekce kostry v prostředí automobilu..



Obrázek 1.26: *Třídní diagram výsledné knihovny*

DriverTracker

Hlavním třídou knihovny pro detekci kostry řidiče v automobilu je *DriverTracker*. Tato třída umožňuje natrénování modelu pozadí pomocí metody *trainBackgroundModel*. Při volání této metody je potřeba, aby se v testovacím prostoru nenacházela žádná osoba, protože metoda zhruba 2s snímá hloubkový obraz scény a udělá její průměrný model, který je následně uložen na disk a při dalším použití knihovny je k dispozici a znovu načten. Metoda *setSteerWheelArea* slouží pro anotaci prostoru kolem volantu a přijímá počáteční pozici, výšku a šířku regionu. Metody *switchToDTDetector* a *switchToHandDetector* slouží pro přepínání mezi dvěmi navrženými metodami pro korekci detekce kostry. První zmíněná metoda využívá metodu hledání loktů a dlaní pomocí transformace vzdálenosti, druhá využívá metodu hledání dlaní v oblasti volantu a detekce překřížených rukou. Pro přidání třídy, která bude přijímat událost o detekovaných gestech slouží metoda *addGestureListener*. Parametrem musí být instance třídy implementující rozhraní *GestureListener*. Pro vykreslení modelu kostry do předaného obrazu můžeme použít *drawSkeleton*. Pro ladění algoritmů lze využít metoda *showDebugTrackbars*. Pro aktuální použitý algoritmus se pak zobrazí okno s posuvníky, které umožní například plynule ladit parametry barevného modelu YCrCb a nastavovat spoustu dalších parametrů ovlivňujících výstupy algoritmu.

Nyní se již dostáváme k poslední a nejdůležitější veřejné metodě třídy. Jedná se o metodu *detect*, která nepřijímá žádné parametry a vráceným výsledkem je objekt kostry obsahující pozice ramen, loktů, dlaní a hlavy. Metoda nejdříve získá z připojeného zařízení Kinect

hloubkový a barevný obraz. V tomto bodě je nutno podotknout, že získané obrazy již máme díky třídě *FreenectUtils* ve formátu *IplImage*, což je typ nesoucí obraz v knihovně OpenCV. Tento obraz je následně zpracován a pomocí načteného obrazu pozadí je ze získaných snímků pozadí odstraněno. Takto upravený hloubkový obraz vstupuje do knihovny Skeltrack, konkrétně je využita funkce synchronní detekce jejíž deklaraci můžeme vidět níže.

```
SkeltrackJointList skeltrack_skeleton_track_joints_sync
                                        (SkeltrackSkeleton *self,
                                        guint16 *buffer,
                                        guint width,
                                        guint height,
                                        GCancellable *cancellable,
                                        GError **error);
```

Parametr *self* je odkazem na vytvářenou kostru a *buffer* je námi zpracovaný hloubkový obraz. Po tom co Skeltrack dokončí detekci je kostra přepracovaná do vlastní reprezentace dané třídou Skeleton a pomocí metody *getHeadWidth* je získaná šířka hlavy. Vytvořená kostra je dále předána spolu s hloubkovým a barevným obrazem aktuálnímu algoritmu pro korekci kostry. Po dokončení korekce jsou detekována základní gesta a notifikováni posluchači těchto gest. Většina gest je detekována podle tvaru kostry. Detekováni otevřené dlaně je realizováno metodou *isOpenHand*. Tato metoda využívá k detekci otevřené dlaně OpenCV funkci findContours a na nalezenou konturu vytvoří konvexní obal pomocí funkce convexHull a defekty pomocí funkce convexityDefects

DTDetector

Třída *DTDetector* realizuje korekci detekce kostry pomocí detekcí dlaní a loktů pomocí aplikace transformace vzdálenosti. Její hlavní metodou je *detect*. Tato metoda přijímá odkazem kostru postavy detekovanou knihovnou Skeltrack, hloubkový a barevný obraz. Kostra je předávaná odkazem, aby mohla být v případě nalezení lepších pozic opravena.

Nejdříve je vytvořen binární obraz kůže pomocí modelu YCrBr, díky metodě *createSkinImage*. Pro převod BGR obrazu do obrazu YCrBr je využita funkce OpenCV - *cvtColor* s parametrem CV_BGR2YCrCb. Pro vylepšení obrazu a odstranění šumu je také aplikován Gaussův filtr. Tento obraz je dále ořezán pomocí metody *cropSkinImage*, aby nám v něm zůstaly pouze body kůže, které se nachází v oblasti volantu a jejichž prostorová pozice je blíž směrem k senzoru od pozic ramen. Pozice ramen jsou získány z předaného objektu kostry. Dalšími kroky jsou vytvoření binárního obrazu osoby a aplikování transformace vzdálenosti. Tyto dva kroky jsou sloučeny v jedné metodě s názvem *createPersonDTImage*. Binární obraz je

vytvářen tak, jak bylo vysvětleno v teoretickém úvodu, aby ve výsledku nebyl zahrnut trup sledované osoby. Postava je díky pozic ramen rozdělena středem na levou a pravou část a body, které jsou v hloubkovém obraze v hloubce konkrétních ramen dané části nejsou zahrnuty. Pro aplikace transformace vzdálenosti je použita metoda *distanceTransform* knihovny OpenCV. Její deklaraci můžeme vidět dále.

```
void distanceTransform(InputArray src,
                       OutputArray dst,
                       int distanceType,
                       int maskSize)
```

Jako *distanceType* bylo použito CV_DIST_L1 a jako *maskSize* CV_DIST_MASK_PRECISE. Bylo totiž vyzkoušeno, že toto nastavení podává pro naši aplikaci nejlepší výsledek. Tento obraz společně s obrazem kůže vstupuje do metody *findArm*, které se ještě předává informace jestli se zkoumá levé nebo pravé rameno. Tato metoda má za úkol nalézt nejlepší kandidáty pro lokty a dlaně. Využívá k tomu metody *findElbowCandidates* a *findHandCandidates*. Tyto metody využívají ke své práci nejspíš nejdůležitější metodu pro tento typ algoritmu a tou je *trackLine*. Tato metoda slouží ke generování přímek a sledování hodnot na této přímce v obrazu transformace vzdálenosti a binárním obrazu kůže. Pro generování přímky je použit Bresenhamův algoritmus, který by měl být efektivní díky používání celočíselných operací. Výsledkem této metody je naplnění struktury *Candidate* naměřenými hodnotami. Tato struktura vypadá následovně.

```
typedef struct {
    float dtValue;
    float dtAccuracy;
    float skinValue;
    int x,y;
    int angleIndex;
    int nonZeroLength;
    int startX,startY;
} Candidate;
```

Hodnota *dtValue* reprezentuje průměrnou hodnotu vzdáleností napříč přímkou, *dtAccuracy* podává informaci o míře pokrytí v procentech. Hodnota *skinValue* určuje množství pixelů kůže na přímce. Hodnoty *startX*, *startY* a *x*, *y* udává počáteční a koncovou souřadnici přímky. Pro identifikaci použitého úhlu pro generování přímky je zde hodnota *angleIndex* a poslední

nonZeroLength určuje délku přímky, která je ještě pokryta body hloubky. Díky těmto hodnotám je algoritmus schopný nalézt konkrétní nejvhodnější přímku pokrývající hledanou oblast, ať se jedná o hledání pozice loktu nebo ruky.

HandDetector

Třída *HandDetector* realizuje korekci detekce kostry pomocí detekcí dlaní v oblasti okolí volantu. Její hlavní metodou je znovu *detect*. Tato metoda přijímá odkazem kostru postavy detekovanou knihovnou Skeltrack, hloubkový a barevný obraz. Nejdříve je znovu vytvořen binární obraz kůže pomocí metody *createSkinImage* a modelu YCrBr, podobně jako v předchozím případě. Opět je zpracována oblast pouze v okolí volantu a v potaz se berou pouze pixely které se nachází od ramen blíž. Po tomto kroku voláme metodu *findHands*, která má za úkol projít v oblasti kolem volantu všechny hloubkové body a větší souvislé oblasti označit. Tyto oblasti se často označují bloby. Tyto oblasti se následně procházejí a porovnávají se jejich parametry, jako plocha a procentuální obsah kůže a jsou vybráni dva nejlepší kandidáti, kteří jsou největší a mají největší procentuální pokrytí kůží. Pokud jsou tyto oblasti nalezeny dvě je oblast nacházející se na levé straně označena jako levá ruka a oblast na pravé označena jako pravá. Pokud jsou nalezeny tyto dvě oblasti je dále volána metoda *detectCrossHands*, která má za úkol rozhodnout jestli uživatel překřížil ruce. Je k tomu využíván binární obraz vytvořený z hloubkového obrazu, kde segmentujeme pouze body nalézající se před rameny. Na tento obraz aplikujeme OpenCv funkci *findContours* a na nalezenou největší konturu vytvoříme konvexní obal pomocí funkce *convexHull* a defekty pomocí funkce *convexityDefects*. Tyto defekty procházíme jsou hledány defekty uvedené v teoretickém rozboru. Pokud jsou tyto defekty nalezené je rozhodnuto, že uživatel překřížil ruce a označení rukou se přehodí. Levá se tímto stane pravou a pravá levou. Po těchto krocích je již provedeno pouze rozhodnutí zdali se nalezené dlaně nalézají v oblasti kolem volantu. Pokud ano je provedena korekce předané kostry a původní parametry dlaní jsou nahrazeny novými.

FreenectUtils

Za povšimnutí stojí určitě i třída *FreenectUtils*. Jedná se o třídu, která plní funkci mostu mezi světem zařízení Kinect a světem OpenCV. Obsahuje čtyři statické metody. Metoda *init* slouží k inicializaci zařízení. Pro inicializaci je potřeba v knihovně libfreenect sekvence těchto tří příkazů.

freenect_init

freenect_num_devices

freenect_open_device

První inicializuje knihovnu a vytvoří *freenect_context* předaný odkazem. Druhý příkaz vrátí počet připojených zařízení a slouží také pro ověření, že je zařízení správně připojeno. Poslední příkaz se snaží zařízení inicializovat a vytvoří objekt *freenect_device* předaný odkazem. Této funkci je také předán index zařízení, což umožňuje pracovat s více zařízeními najednou.

Další metodou je třídy *FreenectUtils* je *freenect_sync_get_rgb_cv*. Jedná se o metodu, která ze zařízení stáhne aktuální barevný obraz a přetvoří jej do typu obrazu, se kterým může pracovat knihovna OpenCV. Využívá k tomu příkaz daný následující deklaraci.

```
int freenect_sync_get_video(void **video,
                            uint32_t *timestamp,
                            int index,
                            freenect_video_format fmt);
```

Příkaz zpracovává získání obrazu synchronním přístupem, proto je také jedním z parametrů i *timestamp*, což je časová známka získání obrazu. Libfreenect totiž v jádru pracuje asynchronním způsobem. Tato časová známka umožňuje zavrhnout starší obrazy, pokud dojde k nějaké desynchronizaci. Parametr index je číslo zařízení a do parametru video je nastavena adresa výsledného obrazu. Tyto data jsou následně nastavena do OpenCV obrazového typu *IplImage* pomocí funkce *cvSetData*. Je třeba si dát pozor protože organizace barevných kanálů v získaném obraze je RGB a OpenCV pracuje v základu s organizací BGR, proto je třeba ještě obraz konvertovat pomocí *cvCvtColor* s parametrem CV_RGB2BGR.

Metoda *freenect_sync_get_rgb_cv* slouží k získání hloubkového obrazu ze zařízení a znovu jej překonvertuje pro použití v OpenCV. Využívá k tomu příkaz daný následující deklarací.

```
int freenect_sync_get_depth(void **depth,
                            uint32_t *timestamp,
                            int index,
                            freenect_depth_format fmt);
```

Princip je v podstatě podobný, jako u příkazu pro získání barevného obrazu.

Poslední metodou je *GlViewColor* a jedná se spíše o metodu pro ladění, protože hloubkový převádí obraz do barevné reprezentace.

6.4 Ladění výsledků knihovny Skeltrack

Už samotným odstraněním pozadí z hloubkového obrazu dojde ke značnému vylepšení výsledků detekce, přesto však je obraz často dosti nestabilní a často se projevuje roztřepáním výsledné kostry. Jelikož je knihovna značně otevřená dává k dispozici značné možnosti ovlivňování výsledků, jejich výčet a částečný popis můžeme najít v [15]. Ve své práci jsem otestoval několik z nich, pouze některé měli pozitivní vliv na výsledky.

smoothing-factor

Hodnota tohoto parametru ovlivňuje míru plynulého přecházení mezi jednotlivými pozicemi pro klouby. Tuto vlastnost je třeba zapnout pomocí *enable-smoothing* parametru. Jedná se o hodnotu mezi 0.0 až 1.0. Čím blíž je tato hodnota nule tím plynulejší výsledky produkuje a kostra produkuje méně efekt třepání kostry. V našem případě však větší míra tohoto zjemňování způsobovala problémy v našem detekčním algoritmu, protože aktuální hodnoty se tímto zjemňováním dostavují až s určitým zpožděním. Pro náš případ byla nalezena nejlepší hodnota výsledků 0.4.

dimension-reduction

Protože je prohledávání prostoru značně časově náročné, používá knihovna redukci prohledávaného obrazu a tento parametr tuto míru redukce ovlivňuje. Tento parametr se pohybuje mezi hodnotou 1 a 1024 a při hodnotě 1 již není použita žádná redukce. V tomto případě je detekce již extrémně pomalá a ani přesto nepodává nějaké velice dobré výsledky. Nejlepší kompromis byl odzkoušen v okolí hodnoty 16. Nižší hodnoty způsobovali enormní degradaci výkonu.

7 Testování

V této kapitole se čtenář dozví o průběhu a výsledcích testů, které byly prováděny na výsledné aplikaci. Byly prováděny dvě úrovně testů. Nejdříve byl navržený algoritmus testován z pohledu výkonosti, aby byla odhalena jeho použitelnost a dále z pohledu funkčnosti, aby bylo možno určit zdali jsou detekovány základní třídy chování postavy u volantu. Celkově byla funkčnost programu testována na pěti lidech, aby se dalo určit zda je funkční i pro jiné postavy a byly porovnány různé typy algoritmů. Také bylo bráno v potaz vliv osvětlení na kvalitu detekce. Výsledky jsou konzultovány a připraveny ke zhodnocení v poslední kapitole a možnosti dalšího směrování

7.1 Testování výkonosti

Díky tomu, že se systém skládá z několika samostatných částí, můžeme měřit dobu trvání odděleně a dále tak hledat nejslabší články systému.

Hlavní důraz na měření výkonu je kladen na metody korekce a odhalit tak nejpomalejší procesy v algoritmech. Také by nám výsledky tohoto testování měli dokázat zda-li systém dokáže plynule pracovat v reálném čase.

První tabulka uvádí celkovou dobu trvání detekce. Je zde uveden zvlášť algoritmus pro detekci kostry pomocí transformace vzdálenosti označen číslicí jedna a také algoritmus hledání dlaní, označen číslem dva. Zvlášť je také uvedena doba trvání samotné knihovny Skeltrack. Jak je vidět druhý algoritmus je ve svém maximu o 58ms rychlejší, dalo se to však čekat, neboť je také principiálně jednodušší. Velice zajímavá je také výkonnost algoritmu Skeltrack, je vidět že je velice dobře optimalizovaný. Od samotné detekce je také nutné odečíst odstranění pozadí, které není uvedeno v tabulce, ale trvá ve svém maximu až 78ms. Rozptyl mezi maximem a minimem u prvního algoritmu je dán tím, že je zde optimalizace, která zapříčiní, že pokud byla na volantu nalezena knihovnou Skeltrack pouze jedna ruka a druhá je v jednom z rohů, korekce se neprovádí. Bylo totiž otestováno, že takovéto situace jsou detekovány správně. Průměrná rychlost detekce kostry u prvního algoritmu se však pohybuje kolem 110ms. To znamená, že zvládá zpracovat až 9 - 10 snímků za sekundu, což není extra plynulé, ale svůj účel zvládá a reakce jsou vcelku adekvátní.

Celková doba

	Čas minimum (ms)	**Čas maximum (ms)**
Detekce kostry 1	77	177
Detekce kostry 2	108	119
Zpracování knihovny Skeltrack	1.74	3.79

V Další tabulce pak můžeme vidět výkonnostní rozbor algoritmu aplikující korekci na základě transformace vzdálenosti. Je vidět, že nejdéle zde trvá vytvoření binárního obrazu kůže. Zpracování binárního obrazu a aplikace transformace vzdálenosti je nad očekávání o něco rychlejší.

Algoritmus aplikace transformace vzdálenosti

Metoda	**Čas minimum (ms)**	**Čas maximum (ms)**
detect	88	109
createSkinImage	48	55
cropSkinImage	9	11
createPersonDTImage	28	31
findElbowCandidates	0.4	0.6
findHandCandidates	0.1	0.9

V poslední tabulce pak najdeme rozbor druhého algoritmu korekce. Je zde vidět, že nejdelší dobu si vybrala metoda hledání dlaní.

Algoritmus hledání dlaní

Metoda	**Čas minimum(ms)**	**Čas maximum(ms)**
detect	27	33
createSkinImage	9	9
findHands	13	15
detectCrossHands	9	9

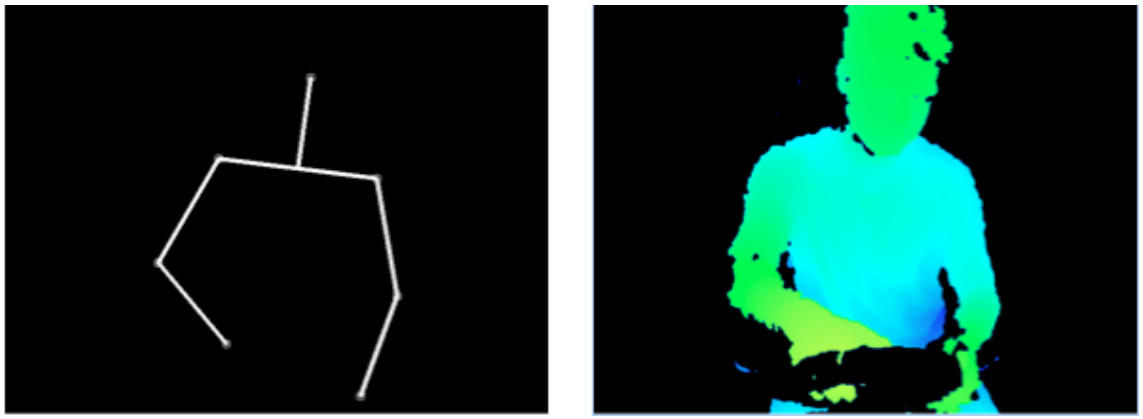
7.2 Testování funkčnosti algoritmu

Bylo testována funkčnost algoritmu při různých situacích a bylo vypozorována spousta zajímavých vlastností algoritmu. Výsledky zajímavých specifických situací lze nalézt na obrázcích níže. Jelikož byla funkčnost testována na pěti lidech, vzniklo spousta zajímavých výstupů programu, které by však tuto kapitolu příliš zahltili. Proto je ještě další série výstupů zobrazená v sekvencích přiložená v příloze na konci tohoto textu.

Nejlepších výsledků podává algoritmus při viditelnosti všech částí těla, toho lze nejlépe dosáhnout zdvihnutím obou rukou. Výsledky se naopak zhoršují pokud je jedna z částí nedokonale viditelná nebo pokud se některá z částí přibližuje příliš k zařízení Kinect. Z hloubkového obrazu pak začínají vypadávat větší části ploch. Toto je nejvíc způsobeno omezenou funkčností zařízení v krátkých vzdálenostech. Pokud se zaměříme na funkčnost v okolí volantu, výsledky bývají většinou přijatelné. Problém přichází v momentech, kdy jedna z rukou zmizí pod volantem, ta je odstraněná díky odstranění pozadí a může se stát, že není viditelná ani část kůže. V této chvíli dochází k problémům u algoritmů založených na hledání kůže a dlaň není vůbec označena. S touto situací se však umí v některých situacích docela dobře vyrovnat algoritmus založený na transformaci vzdálenosti v režimu, kdy nepoužívá obraz kůže. Tuto možnost lze za běhu programu přepínat. Obecně tak lze říct a lze to i vidět ve výsledcích testů zobrazených v přílohách na konci práce, že tento algoritmus bez použití kůže podává nejlepší výsledky a to i za zhoršených světelných podmínek.

Výsledky bývají také o něco méně předvídatelné v momentech prudkých pohybů volantem, někdy dochází ke skokovým změnám. Také naše algoritmy zachovávají jednu z vlastností převzatou z algoritmu knihovny Skeltrack a to, že se kostra v některých momentech jakoby třepe. To je také dáno zčásti nestabilitou hloubkového obrazu v krátkých vzdálenostech a je možné tuto vlastnost omezit snížením parametru smoothnes více popsaném v kapitole implementace. Problémem je, že pak ale náš algoritmus podává o něco horší výsledky, protože se mnohem pomaleji aktualizují pozice ramen. Myslím si však, že tento problém menšího třepání kostry není nikterak velký, aby omezoval použití knihovny.

Na následujícím obrázku lze tedy již vidět výsledky při otáčení volantu. Lze vidět téměř úplně odstraněné široké křeslo a volant. Metoda se v této situaci chová celkem stabilně, pokud neotáčíme volantem příliš prudce a o velké úhly.

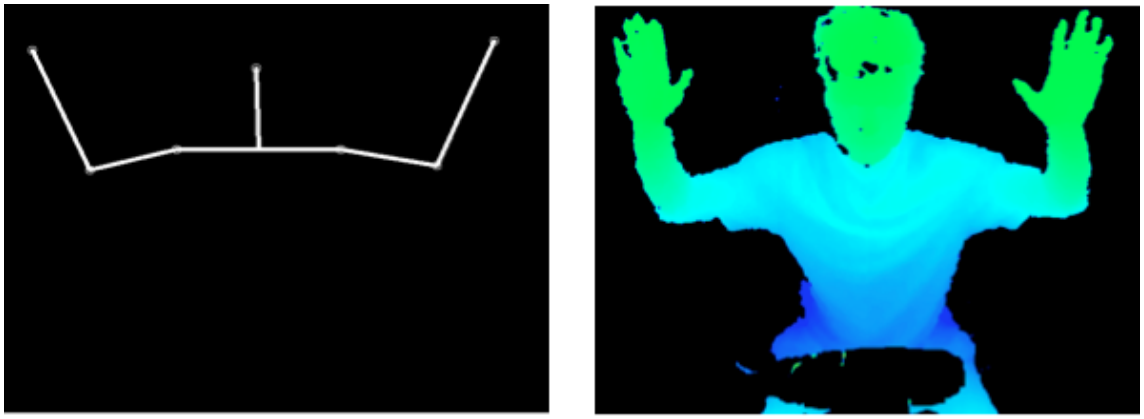


Obrázek 1.27: *Výsledek detekce při otáčení volantu*

Následující obrázek ukazuje chování algoritmu při zkřížení rukou. V tomto momentu se algoritmus chová celkem stabilně a detekce tohoto momentu by měla být bezproblémová.

Obrázek 1.28: *Výsledek detekce při překřížení rukou*

Předposlední ukázka znázorňuje představitele nejstabilnějších detekcí. Algoritmus je schopen bezproblémově klasifikovat situace, kdy jsou lokty a dlaně dobře viditelné. Jedná se například o situace, kdy jsou obě ruce vzhůru. Jedna z rukou vzhůru a druhá na volantu nebo ruce ukazující do spodních rohů a podobné kombinace těchto situací.

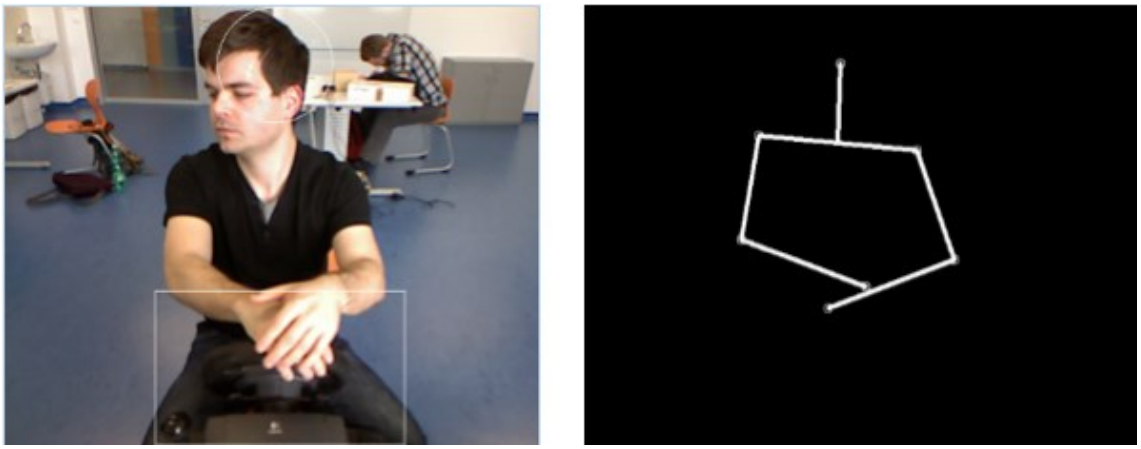


Obrázek 1.29: *Výsledek detece při zvednutí rukou*

A nakonec poslední ukázka představuje situaci, kde jsou již ruce pro algoritmus neviditelné. Tuto situaci nebyl schopen rozpoznat jak Skeltrack, který našel pouze lokty, což lze rozpoznat na obrázku, tak tuto situaci nerozpoznal ani náš algoritmus, kde zachybovala nejspíš detekce kůže. Ta totiž dle návrhu bere v potaz pouze body nacházející se před úrovní ramen.

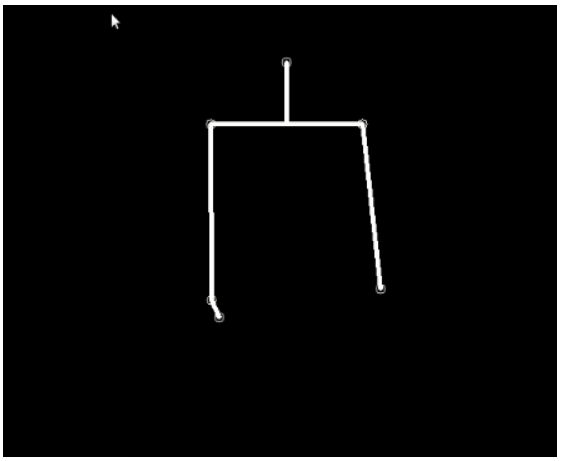
Obrázek 1.30: *Chybná detekce*

Zde je ještě ukázka na barevném výstupu, testována v prostředí školní laboratoře, narozdíl od předešlých, které byly pořízeny v mém bytě.



Obrázek 1.31: *Výsledek detekce*

Je třeba také ještě zdůraznit, že pro představené výsledky byl použit algoritmus detekce loktů a dlaní pomocí transformace vzdálenosti. Algoritmus, který hledá pouze dlaně v okolí volantu často havaroval na výsledcích knihovny Skeltrack, která bohužel často určila špatné pozice loktů. Můžeme to pozorovat na následující ukázce.



Obrázek 1.32: *Výsledek detekce při použití algoritmu pro detekci dlaní v okolí volantu*

Držení volantu je v klasické podobě. Skeltrack však v tomto případě detekoval pravý loket někde blízko pravé dlaně a levý loket nebyl detekován vůbec.

Bylo také vypozorováno, že proces vytváření binárního obrazu kůže je choulostivý na intenzivní umělé osvětlení. V této chvíli byly o něco více reflexivnější povrchy zahrnuty do binárního obrazu kůže. Tato situace lze za běhu ošetřit změněním parametrů YCrBr modelu,

konkrétně hodnotou Y. Přesto je v té chvíli binární obraz kůže docela nekvalitní. V reálné aplikaci si dokážu představit, že by byli parametry YCrBr založeny na profilech a v případě, že by se v automobilu rozsvítilo světlo, tak by se nastavili hodnoty YCrBr modelu do hodnot určených konkrétně pro tuto situaci. Konkrétní ukázku binárního obrazu po rozsvícení všech světel v místnosti lze nalézt dále.

Obrázek 1.33: *Chybná detekce kůže*

Testování detekce gest

Testování detekce gest bylo bráno jako okrajové zaměření práce a sloužilo k nastínění využitelnosti práce. Bylo pro otestováno implementací třídy *GestureListenerImpl*, ta vypisuje do standartního výstupu informace o detekovaných gestech. Detekování otevřené a zavřené dlaně probíhalo celkem bezproblémově. V této oblasti lze vidět velká výhoda hloubkového obrazu. Segmentování dlaně v hloubkovém obraze, pokud byla detekována její pozice, je velmi kvalitní a díky tomu pracuje kvalitně i práce s konturami knihovny OpenCV a označení jestli se jedná o uzavřenou dlaň nebo otevřenou. Kvalita detekce gesta položení rukou na volant záleží od výsledku kvality detekce kostry osoby. Využitelnost by musela být ověřena časem. Osobně jsem pociťoval lehkou únavu ruky při jejím držení ve vzpřímené poloze a dokázal bych si představit využití nějakých krátkých gest, jako potvrzení nějaké akce zavřením a otevřením dlaně.

Závěr

Na této práci bylo pracováno s cílem vytvořit systém, který by mohl položit základy pro aplikaci uplatnitelnou v reálném životě a pokud možno zlepšila komfort řidiče automobilu nebo jeho bezpečnost. Výsledky vygenerované kostry by mohly být použity pro ovládání příslušenství automobilu specifickými gesty. Kostra by mohla být také použita pro analýzu pohybu řidiče a jeho držení volantu, s mírnou úpravou i natočení hlavy a směr pohledu a tak analyzovat bezpečnostní anomálie.

Úvodní nadšení však vystřídaly obavy z přílišné komplikovanosti daného problému. Současné práce jsou příliš zaměřené na analýzu samostatných osob, většinou ve vzpřímené poloze a také často zaměřené na herní průmysl a všechny dostupné implementace selhávaly na analýze sedícího člověka při doteku s volantem. Vydal jsem se tedy cestou korekce algoritmu.

Mé největší starosti plynuly z obavy, že přidané algoritmy způsobí citelný výkonnostní propad celé aplikace. Dle výsledků výkonnostních testů se toto potvrdilo částečně, ale aplikace je stále ještě použitelná v reálném čase. Navíc je v aplikaci pořád ještě prostor k optimalizaci, protože dle mého názoru by šlo většinu operací převést do mnohem méně průchodů. Algoritmus je totiž kvůli názornosti značně rozčleněn. Také lze výkon určitě vylepšit paralelním zpracováním.

Nejdůležitějším parametrem zhodnocení jsou však výsledky detekce kostry a testy ukázaly, že na tomto poli je ještě práce pro vylepšování. Dle mého názoru je však možno s vynaloženým úsilím použít navržený systém pro kvalitní detekci, jak některých gest, tak zvyšovat bezpečnost v automobilu, sledováním pozic částí těla a regulovat tak výkon airbagů. Systém by se dalo použít pro ovládání některých palubních systému automobilu nebo navigace. Navrhnutí sady gest bylo také součástí práce. Myslím, že by se systém dalo také použít k analýze pohybů řidiče a na jejich základě provést nějakou akci, podobně jako v systému prototypu Toyoty uvedeném v kapitole aktuální stav.

Možnou výhodou a taky možností výsledky vylepšit je díky návrhu a členění aplikace. Použití korekce je možno uplatnit na jinou a kvalitnější knihovnu na detekci kostry pro zařízení Kinect. Problémem současného stavu je však, že hodně dostupných nástrojů potřebuje ke své inicializaci, aby uživatel provedl kalibrační pózu. Problémem je také, že většina knihoven je značně uzavřená a není příliš možností je upravit ke svému použití.

Podařilo se mi v závěru naimplementovat funkční a použitelný algoritmus pro detekci loktů a dlaní, který už v závěru využíval jen pozic ramen a hlavy získaných z knihovny Skeltrack. Tato zkutečnost se však stala částečně limitující, protože v některých hraničních případech Skeltrack nepodával příliš dobré výsledky. Je tedy otázkou jestli by nebylo lepší pro detekci, třeba pouze hlavy, vyzkoušet jinou z dostupných knihoven.

Pokud bych měl však na základě zkušeností definovat cestu kterou se vydat, aby aplikace podávala opravdu kvalitní výsledky, zvolil bych jednu z metod strojového učení. Tato metoda je však značně náročná a to již z toho důvodu, že neexistuje ověřená trénovací množina.

Některé metody používají pro tuto oblast vlastní generátor syntetických dat, který by bylo potřeba také vymyslet. Je také otázkou zdali by tato množina měla počítat z objekty nacházející se v okolí řidiče. Tvary těchto objektů mohou být specifické pro jednotlivé typy automobilu a jak odhalila zkušenost, odstraněním pozadí můžeme přijít o některé důležité informace a určitě to nevyřeší případ řidiče, který je připoután bezpečnostními pásy.

Použitá literatura

[1] PFLEGING, Bastian, Tanja DÖRING, Martin KNOBEL a Albrecht SCHMIDT. AutoNUI: A Workshop on Automotive Natural User Interfaces. 2011.

[2] GREENE, Jay. Behold: The Microsoft Mustang!. Dostupné z: http://www.cnet.com/pictures/behold-the-microsoft-mustang-photos/4/

[3] SILBERT, Sarah. Toyota's Smart Insect concept EV packs Kinect motion sensor, voice recognition. Dostupné z: http://www.engadget.com/2012/10/02/toyotas-smart-insect-concept-ev-kinect/?utm_medium=referral&utm_source=pulsenews

[4] BRANDRICK, Chris. Kinect Hack Helps You Park Your Car, Can't Help Drive It Yet. Dostupné z: http://www.techhive.com/article/249986/kinect_hack_helps_you_park_your_car_cant_help_drive_it_yet.html

[5] Kinect. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-05-04]. Dostupné z: http://en.wikipedia.org/wiki/Kinect

[6] SIDDIQUI, Matheen a Gerard MEDIONI. Human Pose Estimation from a Single View Point, Real-Time Range Sensor [online]. Los Angeles [cit. 2014-05-04]. University of Southern California.

[7] SHOTTON, Jamie, Andrew FITZGIBBON, Mat COOK, Toby SHARP, Mark FINOCCHIO, Richard MOORE, Alex KIPMAN a Andrew BLAKE. Real-Time Human Pose Recognition in Parts from Single Depth Images [online]. [cit. 2014-05-04]. Dostupné z: http://research.microsoft.com/pubs/145347/bodypartrecognition.pdf. Microsoft Research Cambridge.

[8] ABHISHEK, Kar. Skeletal Tracking using Microsoft Kinect [online]. [cit. 2014-05-04]. IIT Kanpur.

[9] OpenNI. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-05-04]. Dostupné z: http://en.wikipedia.org/wiki/OpenNI

[10] Prime SensorTM NITE 1.3 Algorithms notes [online]. PrimeSense, 2010 [cit. 2014-05-04]. Dostupné z: http://pr.cs.cornell.edu/humanactivities/data/NITE.pdf

[11] MICROSOFT. Kinect for Windows Architecture [online]. [cit. 2014-05-04]. Dostupné z: http://msdn.microsoft.com/en-us/library/jj131023.aspx

[12] BAAK, Andreas, Meinard MULLER, Gaurav BHARAJ, Hans-Peter SEIDEL a Christian THEOBALT. A Data-Driven Approach for Real-Time Full Body Pose Reconstruction from a Depth Camera. [online]. [cit. 2014-05-04]. Dostupné z: http://www.mpi-inf.mpg.de/~abaak/download/2011_BaakMuBhSeTh_DataDrivenDepthTracking_ICCV.pdf

[13] Distance transform. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-05-04]. Dostupné z: http://en.wikipedia.org/wiki/Distance_transform

[14] DAVIES, Chris. Microsoft Connected Car plans include Kinect, WP8 and the cloud. Dostupné z: http://www.slashgear.com/microsoft-connected-car-plans-include-kinect-wp8-and-the-cloud-25235418/

[15] ROCHA, Joaquim. Announcing Skeltrack [online]. 2012 [cit. 2014-05-04]. Dostupné z: http://www.joaquimrocha.com/2012/03/21/announcing-skeltrack/

[16] BURRUS, Nicolas. Kinect Calibration [online]. [cit. 2014-05-05]. Dostupné z: http://nicolas.burrus.name/index.php/Research/KinectCalibration#tocLink0

[17] ANDÚJAR, C. Kinect [online]. 2012 [cit. 2014-05-04]. Dostupné z: http://www.lsi.upc.edu/~virtual/RVA/Course%20Slides/Kinect.pdf

Seznam příloh

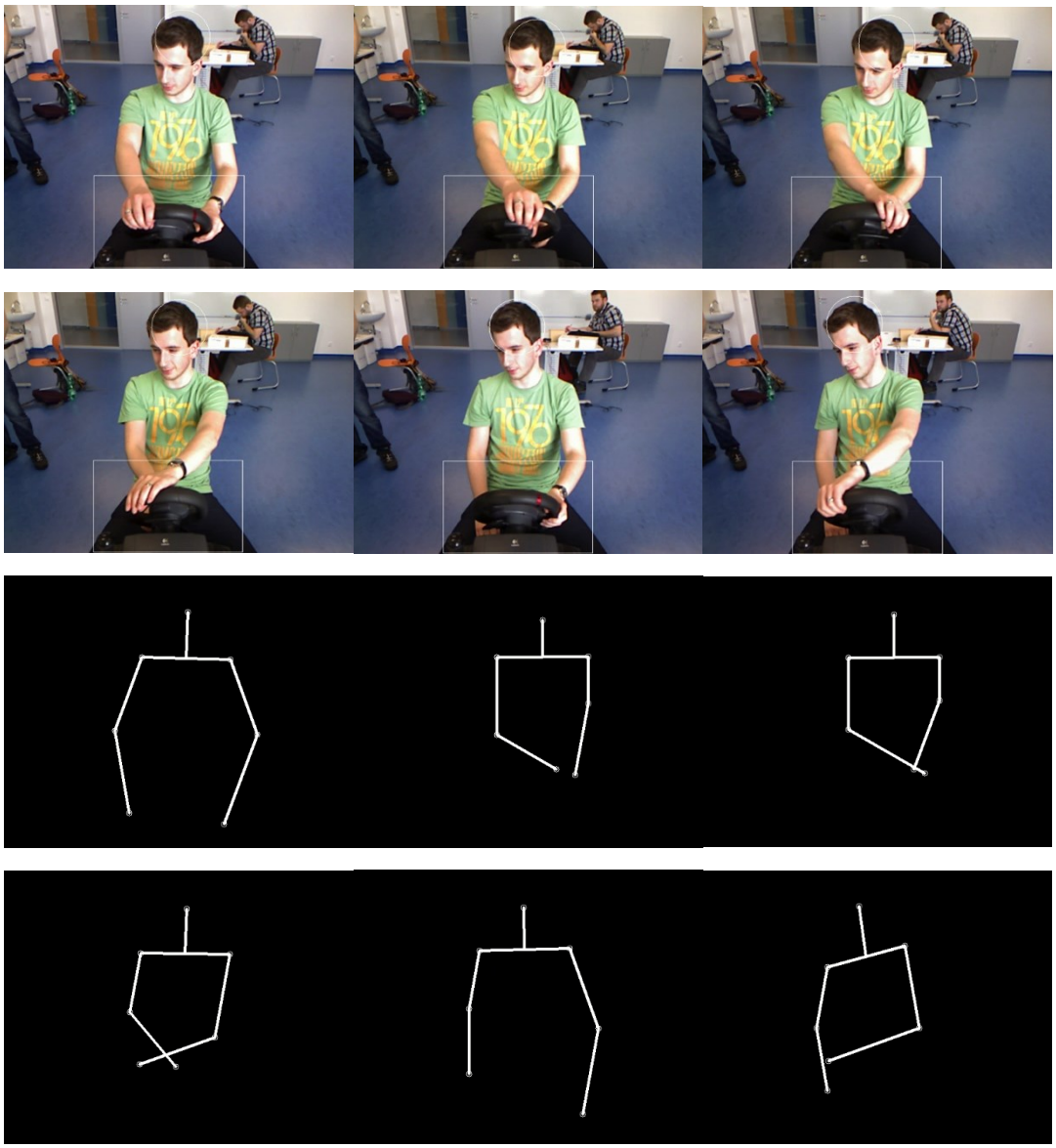
Příloha A: Výsledky testů

Součástí diplomové práce je CD.

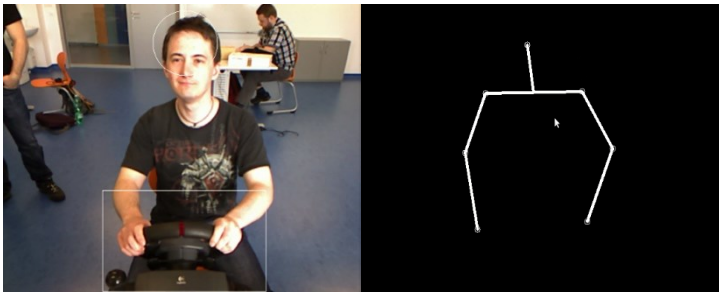
Na disku je uložen zdrojový kód programu, návod na zprovoznění knihovny Skeltrack a také videozáznam z testování.

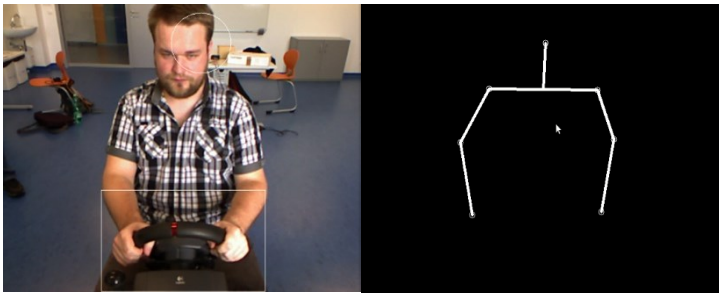
Příloha A: *Výsledky testů*

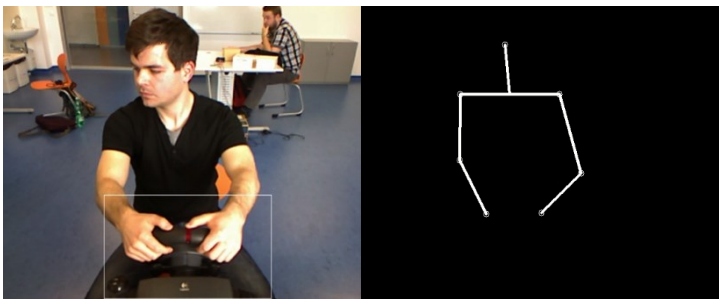
Průběh točení volantem



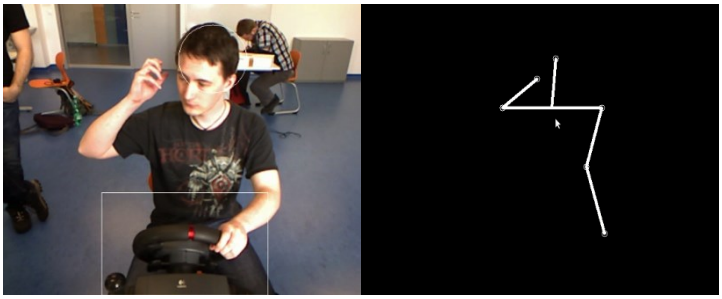
Základní pózy



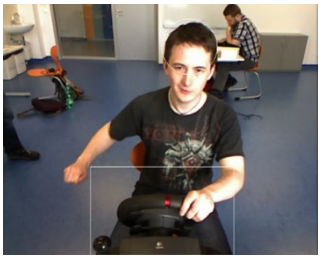


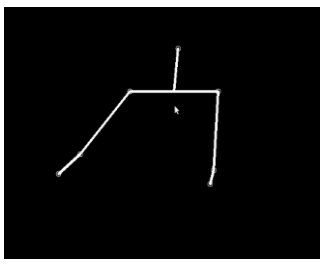


Póza „volajícího“

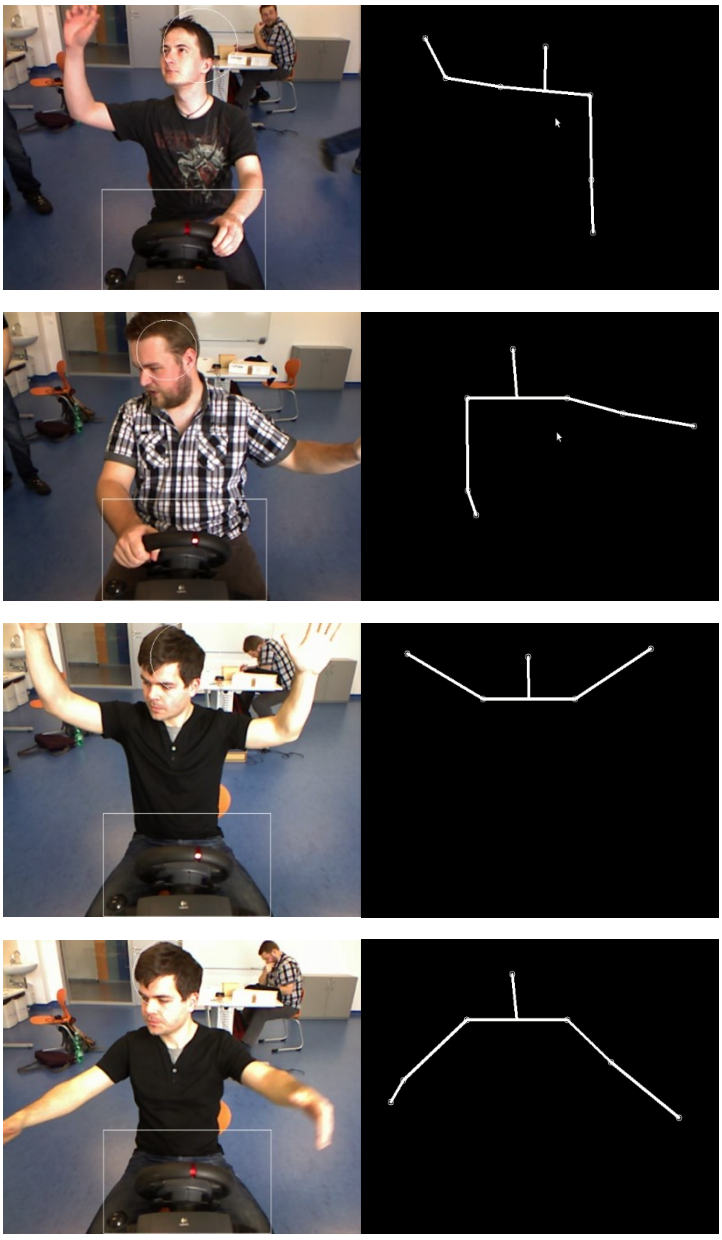


Póza „řadící páka“

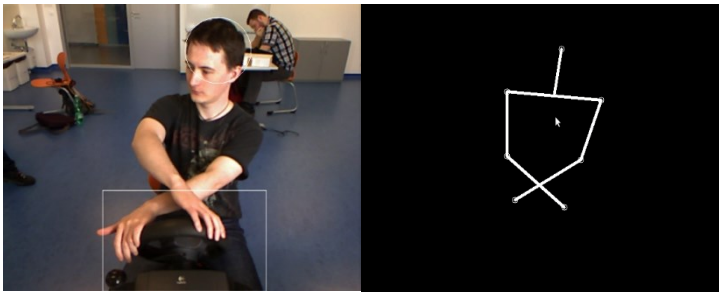


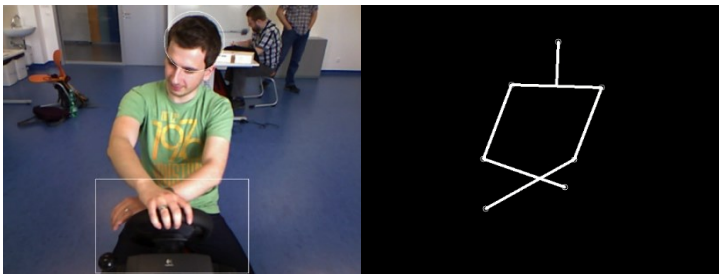


Ruce mimo volant

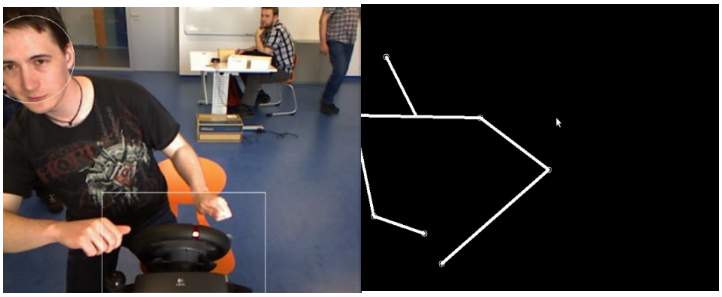


Překřížené ruce





Výstup z auta



Neklasifikované pózy